POPmessage

App de missatgeria amb geolocalització

Autor: Albert Costas Gutiérrez
Grau d'Enginyeria Informàtica
Java EE

Consultor: Santi Caballe Llobet
Professor responsable: Albert Grau Perisé

Data Lliurament: 14 de juny del 2017

FITXA DEL TREBALL FINAL

Títol del treball:	*Aplicació de missatgeria amb geolocalització*
Nom de l'autor:	*Albert Costas Gutierrez*
Nom del consultor/a:	*Albert Grau Perisé*
Nom del PRA:	*Santi Caballe Llobet*
Data de lliurament (mm/aaaa):	*05/2017*
Titulació o programa:	*Grau d'Enginyeria Informàtica*
Àrea del Treball Final:	*JAVA EE*
Idioma del treball:	*Català*
Paraules clau	*Missatges, Geolocalització, Votació*

Resum del Treball (màxim 250 paraules):

En la actualitat, dins de l'àmbit de les noves tecnologies, la inclusió del telèfons intel·ligents i l'aparició de les xarxes socials, ha creat una **interconnexió entre les persones amb temps reals**.

En aquest sentit, el projecte **intentara interconnectar persones mitjanant missatges els quals estaran visibles per tots aquells usuaris en el mateix punt geogràfic**.

S'ha realitzat un programari de dispositiu mòbil, el quan proposa d'una manera **senzilla enviar un missatge en l'àrea la qual es troba l'usuari**. I altrament, de poder veure els missatges mes valorats, amb la possibilitat de **votar positivament o negativament**.

La part d'iteració del usuari és una aplicació de **dispositiu mòbil que consumeix una API amb JAVA EE (Jersey & Jackson)**, que subministrara tant la informació necessària així com el punts d'entrada per poder publicar.

L'API disposa d'una **base de dades distribuïda amb un clúster de nodes** per tenir alta disponibilitat. S'han utilitzat comandes (**command pattern**) de lectura/escriptura, aquestes utilitzen les **interfícies** o contractes del repositoris d'accés a dades (**repository pattern**).

La metodologia usada, ha sigut **BDD** amb la definició dels casos d'ús, a partir d'aquests s'han realitzar tasques que mitjançant **metodologia Kanban** s'han executat.

En conclusió, és una aplicació que permet la comunicació esporàdica d'usuaris per la seva situació geogràfica, sigui per veure altres opinions, comentaris, indicacions,... que m'ha permès aprendre a realitzar aplicacions per dispositius mòbils, API's i la seva seguretat, així com una arquitectura escalable i distribuïble. A més, saber com

aprovisionar l'entorn i la seva publicació mitjançant Docker i Docker-Compose.

Abstract (in English, 250 words or less):

In the actuality, in the field of the new technologies, the inclusion of the intelligent telephones and the apparition of the social networks, has created one interconnection between people with real time.

In this sense, the project tried to interconnect people with messages which will be visible for all those users in the same geographic point.

It has realized a software of mobile device, the when proposes way send a message in the area which finds the user. And otherwise, to see the messages month valued, with the possibility to value positively or negatively.

The front-end application is to app mobile device that consumes to API with JAVA EE, that supplied so much the necessary information, as well as the points to begin with for can publish.

The API has of a database distributed with a cluster with nodes to have high availability. They have used commands of reading/writing, these use the interfaces of the repositories for access to data information.

The methodology used, has been BDD with the definition of the cases of use, from these have realize tasks that by means of methodology Kanban have executed for order.

In conclusion, the application that allow the sporadic communication of users for his geographic situation, was to see other opinions, comments,... That has allowed me learn to realize applications for mobile devices, API's and his security, as well as an architecture scalable and distributed. Besides, know how provisioner the environment and his publication with Docker and Docker-Compose technologies.

Treball de final de Grau (TFG)
Àrea J2EE
Projecte POPmessage
App de missatgeria de geolocalització

Índex de continguts

1. Descripció de l'aplicació

En la actualitat, dins de l'àmbit de les noves tecnologies, la inclusió del telèfons intel·ligents i l'aparició de les xarxes socials, ha creat una interconnexió en temps reals mes enllaç, fins i tot, del correu electrònic.

Aplicacions de missatgeria em temps real, són d'ús quotidià, des de l'aplicació que ens envia un missatge senzill de persona a persona, o les anomenades xarxes socials on un grup de persones, o fins i tot, qualsevol persona a nivell mundial, pot veure que un missatge de 140 caràcters, una imatge, un vídeo, etc..

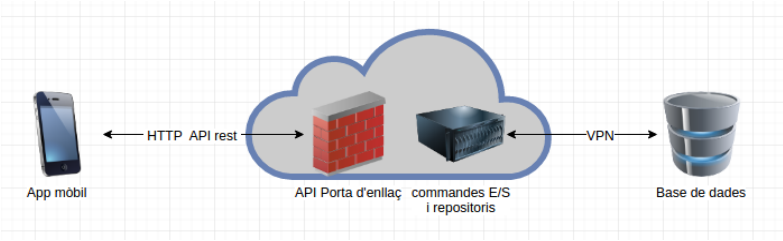
En aquest sentit, l'aplicació que es realitzara en aquest projecte, precisament, intentara interconnectar persones mitjanant petits missatges els quals estaran visibles per tots aquells usuaris en el mateix punt geogràfic.

El programari consisteix, en la part d'iteració usuari-aplicació, en una programari de dispositiu mòbil, el quan proposa d'una manera senzilla i fàcil la possibilitat d'enviar un missatge en l'àrea la qual es troba.

I altrament, de veure aquells missatges mes valorats, amb la possibilitat de valorar positivament o negativament qualsevol d'aquests, de la zona on es troba el mateix usuari.

L'arquitectura de l'aplicació serà una aplicació de dispositiu mòbil, client-servidor, on disposarem un programa java (Jersey & Jackson) que efectuara de servidor, és una API com a porta d'enllaç, que subministrara tant la informació necessària, així com el punts d'entrada per poder publicar aquests missatges. La API utilitzara una serie de comandes (command pattern) de lectura/escriptura. Les comandes utilitzaran la interfases del repositoris d'accés a dades (repository pattern).

Aquesta informació es persistir-ha en una base de dades especialment destinada aquest tipus de documents, base de dades documentals o de grafs. En tot cas aquesta serà de programari lliure (Elastisearch, neo4j,...).



Durant la etapa de disseny, la definició del projecte es realitzarà amb metodologia BDD, plantejant un disseny a partir del casos d'ús durant les etapes inicials del projectes. Aquestes es transformaran en tasques que es realitzar-han per aconseguir els objectius del treball.

Finalment, en l'etapa de execució i per l'entorn de desenvolupament, es realitzarà amb contenidors portables de programari que s'executaran independentment del sistema operatiu (docker) amb un sistema per aprovisionar l'entorn de treball (docker composer, ansible,...). Per al codi i el seu manteniment s'utilitzara un repositori de codi (github, bitbucket,...)

2. Objectius generals i específics

En aquest apart definirem, per una part, els objectius tant a nivell de treball final de grau, i per altra, aquells relatius al programari.

2.1 Objectius TFG

Com a estudiant del grau d'enginyeria informàtica els principals objectius assolir són:

- Adquirir domini i formalitzar-me amb la realització d'aplicacions per a dispositius mòbil amb SDK Android® .
- Adquirir domini i formalitzar-me amb la tecnologia Java EE amb orientació a objectes, i fer servir els estàndards de desenvolupament d'aplicacions amb eines de programari lliure.
- Implementar patrons de disseny, així com una arquitectura escalabre amb interacció d'usuari (capa de presentació), API de porta d'enllaç encarregada d'autenticació d'usuari i talla focs, i la capa de persistència o base de dades.
- Ampliar coneixement d'una aplicació escalable, *desacoplada* amb capes, i tenir present el teorema de CAP, on no podem garantir simultàniament les tres característiques d'una aplicació distribuïda.

2.2 Objectius general del programari

Pel que fa el programari, com extraem de la anterior presentació del projecte, els objectius generals són:

- La interfície d'usuària ha de ser senzilla i simple. La lectura, opinió i creació de missatges s'ha de fer una forma intuïtiva.

- Es necessari una autenticació per usar l'aplicació (nom usuari i contrasenya), el registre del usuari ha de ser amb el mínim de dades necessaris, a model de exemple: nom complet, any de naixement, nom d'usuari i contrasenya.
- Les dades han d'estar emmagatzemades amb total confidencialitat, centralitzades en una base de dades, i des de on es podran fer copies de seguretat. La contrasenya no s'ha de poder veure mai en format pla, aquestes s'enregistraran codificades.

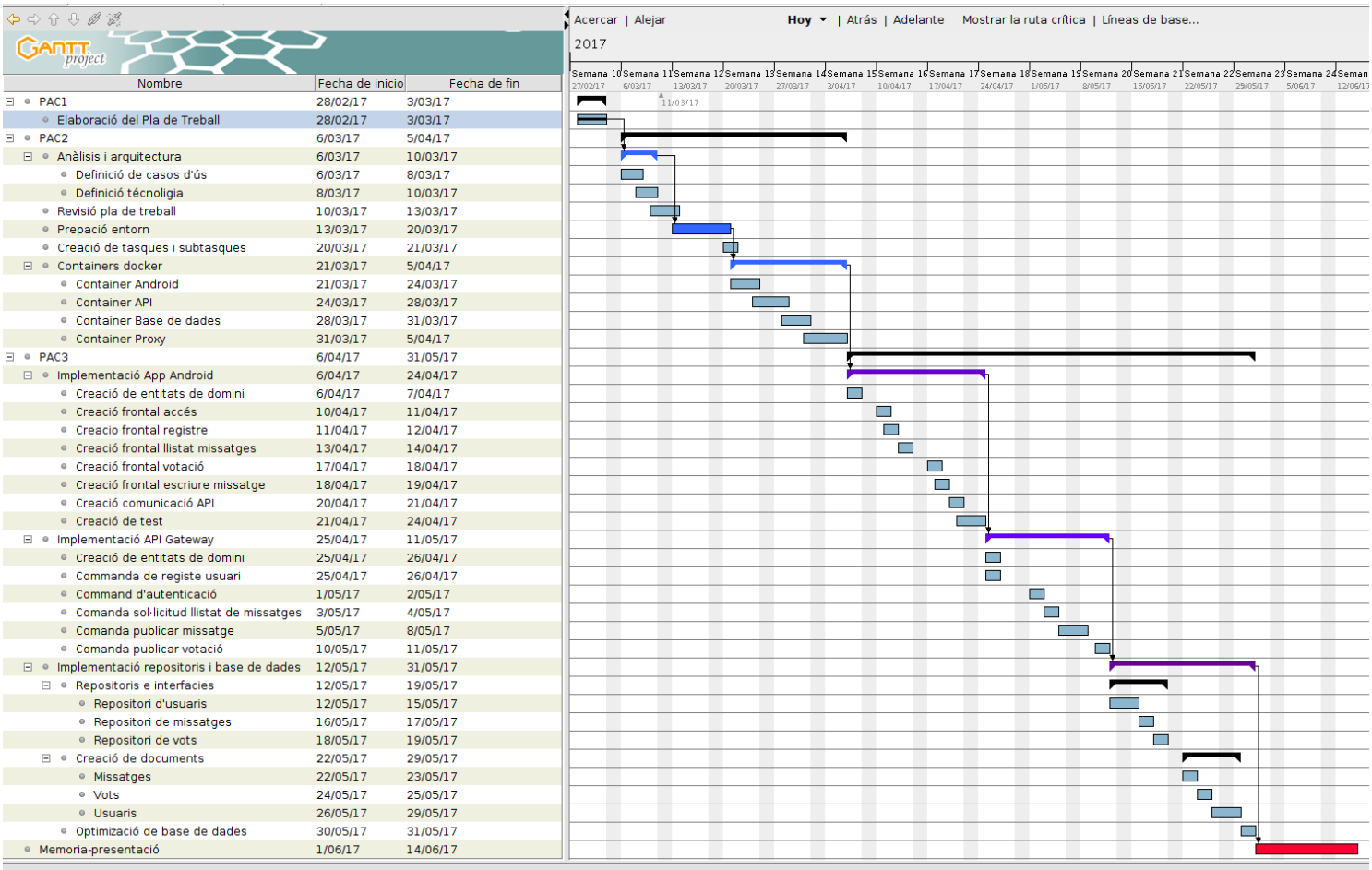
2.3 Objectius específics

- En el moment que és carregui el programari, d'una forma automàtica, s'han de veure els últims missatges d'un radi geolocalitzat que es definira per defecte, i aquesta zona variara en grandària depenen de la quantitat de missatges mínims a mostrar. És a dir, s'ordenara el contingut per la seva puntuació i per la distancia de la seva creació al punt de la sol·licitat.
- Qualsevol missatge que visualitzi l'usuari el podrà valorar positivament o negativament.
- Una icona perfectament visible en tot moment s'utilitzara per escriure un missatge ràpidament i enviar-se amb la localització del dispositiu mòbil, amb un màxim predefinit de missatges per usuari i zona.

3. Planificació del desenvolupament

Planificarem la feina per fer petites entregues o «checkpoints», per realitzar i executar els diferents objectius del projecte, aquestes es basen amb les dates d'entrega de les PAC's.

Podem veure al diagrama les dates i el camí a seguir per cada practica d'avaluació continua, aquestes són un objectiu clar que està format per petites tasques o subtasques.



Veure amb detall:
[http://storage4.static.itmages.com/i/17/0308/h_1489012061_5139705_dad9e8179e.png]

4. Requeriments

L'anàlisi de requeriments s'ha de centrar amb les següents primícies:

- **L'usuari es l'eix al voltant que es dissenya la interfície** de l'aplicació mòbil, aquesta ha de ser intuïtiva, és a dir, ha de ser usable sense necessitat d'informació addicional.
- El sistema ha de proporcionar un mecanisme per **autentificar i autoritzar** usuaris, per a les publicacions, lectures o les votacions.
- S'han d'utilitzar **programari lliure**, amb una **comunitat** que mantingui el projecte lliure, així com ser artefactes **provats** que solvatin el problema.
- Implementar patrons de disseny, així com una **arquitectura per capes i escalabre:**
    - **Capa de presentació:** La interacció d'usuari en aquest cas serà una aplicació mòbil, però en un futur podria ser una web.
    - **Capa de control: API de porta d'enllaç** encarregada d'autenticació d'usuari i talla focs per accedir aquesta capa. Aquesta accedira a una serie de **comandes (command pattern)** per llegir/escriure mitjançat **interfases de repositoris (repository pattern)** que donaran accés a la **capa de persistència**, aquest fet en garanteix poder canviar de base de dades en un futur.
    - **Capa de persistència**: La base de dades ha de reflectir el model conceptual i operatiu on els usuaris te les dades i missatges.

5. Anàlisi

5.1 Introducció

Per poder definir un disseny de la solució al problema em d'analitzar de manera formal els requeriments.

Per tant, primer veurem el context del projecte, tant l'entorn del mateix com les parts implicades.

Aquest primer fet, també en permetrà continua amb la viabilitat del projecte així com la seva capitalització econòmica, les possibilitats per monetitzar i quina etapa de la vida del explicatiu seria possible.

Amb els context i parts implicades detectades, podrem crear els model de casos d'ús i actors que aportin valor.

Finalment, ja podrem crear les fitxes de casos d'ús, casos especialment complexos que ens permetran crear les tasques d'implementació, amb els actors i les internacions amb el sistema, d'una forma molt més explicita.

5.2 Context en l'actualitat

Ems hem de situar dins de l'àmbit de les noves tecnologies, especialment amb la inclusió del telèfons intel·ligents i l'aparició de les xarxes socials, les quals han creat una interconnexió en temps reals dels seus usuaris.

Aproximadament el 86% de les persones que accedeixen a internet és mitjançant el telefon intel·ligent, d'aquestes el 90% usa les xarxes socials gairebé **a diari,** i la suma setmanal d'hores d'ús es d'una mitjana de 3 hores:

«En el puesto número uno en cuanto a la red más usada y visitada está la creada y llevada a la gran pantalla, Facebook, que es usada por el 96% de la población, seguida de Youtube, con un 66% de audiencia, Twitter, 56% de usuarios y Google + con un 34%...

... Calculado por horas cada siete días Facebook acumula 4 horas y media, Youtube 3 horas y media, Twitter 9 horas e Instagram 2 horas y 57 minutos... »

Forbes.es - **CHRISTIAN RODRÍGUEZ** - 3 julios 2015

Si ens enfoquem en les aplicacions de missatgeria actuals WhatsApp®, Telegram®, Line®, Foursquare®,... per similitud amb el projecte, el principal factor comú es la

comunicació directa entre usuaris que prèviament han de compartir el numero telefònic.

Com a diferencia important, WhatsApp® s'està orienta cap la integració en la xarxa social de Facebook® i Instagram®. Per altra banda, Telegram® es diferencia per tenir una comunitat de creació de bots [https://telegram.org/blog/bot-revolution] o automatització de processos mitjançant la seva API.

5.2.1 Monetarització de les xarxes socials

En centrarem amb la monetarització de les grans xarxes socials amb la publicitat pel seu caràcter innovador, tal com veurem.

Aquestes xarxes ens ofereixen plataformes (obertes) o «marketing api's» (sols per empreses col·laboradores) per la creació de creativitats dedicades la publicitat, on termes com cpc* o cpm* indica al publicista com es distribuïx la seva inversió en publicitat.

Fet el qual, es creen creativitats publicitaries a partir de catàlegs de productes, vídeos, imatges,... , o la combinació de tots aquests, amb la capacitat d'arribar a un «targeting» o segment de població: posició geogràfica, genere, gustos musicals, afinitats d'oci,... permeten una **conversió** molt alta, es a dir, una **gran efectivitat entre el cost de publicitat i el retorn econòmic que s'obté**.

Finalment, aquestes milions d'usuaris que permeten aquesta **segmentació** en la publicitat i enfocar-la al perfil desitjat (gairebé impossible en qualsevol altre medi publicitari) se li pot afegir valor amb el **Deep learning** [https://support.google.com/adwords/answer/190596?hl=en] per tenir gairebé campanyes optimitzades em temps reals per **eficiència publicitaria**, alguns exemples:

- https://support.google.com/adwords/answer/190596?hl=en
- https://www.facebook.com/business/learn/facebook-create-ad-dynamic-ads

Finalment nivell organitzatiu, la utilització de «parners» o empreses col·laborades a nivell local o estatal, per la difusió i adaptació d'aquestes tipus de productes, és mes una estratègia de gran corporació per la penetració i personalització en mercats locals.

---

* **cpc**: El cost per clic és un mecanisme de compra de publicitat molt atractiu i rendible para determinats objectius de màrqueting. En el model de CPC l'anunciant no paga en funció de l'audiència que veu un anunci, sinó en funció de l'usuari que respon a l'anunci, realitzant un clic i manifestant així el seu interès a visitar la web de l'anunciant per saber més.

* **cpm**: Cost per impressió equivalent el cpc però en aquest si es paga per impressió de la publicitat.

5.3 Viabilitat del projecte

Els criteris de viabilitat d'un producte d'enginyeria el podem dividir en quatre àrees: operativa, tècnica, dates d'entrega i economia.

5.3.1 Viabilitat operativa

El criteri de viabilitat operativa es mesura per la urgència del problema o l'acceptació de la solució, en aquest cas estem davant d'un projecte final de grau i no en la priorització de projectes d'una entitat o empresa. D'altra banda podem respondre igualment a les preguntes:

- ¿Mereix la pena resoldre el problema o funcionara la solució?
    - Si, la preparació i implementació d'una projecte final de grau ens aparta experiència, bones practiques i formació sobre projectes professionals.
- ¿Que opinen els usuàries finals i els directius sobre la solució?
    - No podem contestar directament, d'altra banda l'estudi de l'entorn i de les parts implicades ens ha de permetre generar o establir objectius, per poder veure l'evolució en la vida del programari així com els punts clau per considerar-ho un cas èxit.

5.3.2 Viabilitat tècnica

La viabilitat tècnica s'ha de fer paral·lelament amb la definició de l'anàlisis, és a dir, sols podem avaluar la viabilitat tècnica en les fases que hi hagi que resoldre qüestions tècniques.

Tot i això, si podem fer una valoració inicial a partir dels primers requeriments, així com dels recursos disponibles i el pla de treball.

Risc de desenvolupament:

En el pla de treball s'indica una proporció major de temps en el desenvolupament de la primera etapa, el desenvolupament de l'aplicació Android©, tant per desconeixent com per complexitat, a més, es la capa de presentació on haurem de realitzar l'estudi d'usabilitat.

Tant mateix, en les primeres etapes s'ha de realitzar la preparació de l'entorn desenvolupament.

Per altra banda, per disminuir el risc hem de tenir present les limitacions indicades en els requeriments com l'arquitectura de l'aplicació.

Disponibilitat de recursos:

Els **recursos són molt limitats**, un **risc elevat** degut a que sols disposem d'un recurs, que s'haurà adaptar a les previsions del pla de treball, definirem les tasques clarament (metodologia BDD) i delimitades per la seva realització, amb els casos d'ús.

Tecnologia:

¿Ha evolucionat la tecnologia fins un punt que sigui possible suportar el sistema? La valoració de les tecnologies ha de complir els requeriment indicats en el punts anterior: han de ser programari lliure, tecnologies amb una gran comunitat i artefactes provades que solucionen el problema:

- Aplicació Android®, pel seu **inici** com a programari lliure i per ser el sistema operatiu mes utilitzat als dispositius mòbils.
- L'API, capa de control, amb JAVA®, com a servei sota demanda.
- Base de dades documental (noSql), con Elasticsearch® que ens soluciona d'una forma eficient la alta disponibilitat, gestió de punts geogràfics i agregats.

5.3.3 Viabilitat dates entrega

S'ha realitzat en el pla de treball, punt 3, on podem veure la gestió del primeres fases, on hi ha el major risc. Es crearan les tasques amb els casos d'ús, limitant l'aplicació i prioritzant requeriments, per poder aconseguir les dates d'entrega.

Important realitzar el projecte amb **metodologia Àgil** per poder realitzar **iterativament (millores incrementals)** el producte, en tot cas, s'inclouran en la memòria del projecte les millores i les funcionalitats que no s'ha pogut realitzar en l'abast i temps establer.

5.3.4 Viabilitat econòmica

L'estudi de viabilitat econòmica ha d'estudiar la justificació econòmica, el risc tècnic i els problemes legals.

Els **risc tècnic es baix**, tot i la manca de recursos per ser un TFG, per contra, la tecnologia empleada està provada així com el patrons indicats.

Per altra banda, si hem de afrontar **problemes legals**, fet que haurem d'indicar clarament amb les clàusules d'us del programari, encara que les dades d'usuari siguin

mínimes per part de l'usuari, s'ha de complir la **LOPD**, especialment per la protecció de dades, indicar l'acotació i el propòsit de l'aplicació, i informar a l'usuari sobre els seus drets. [http://ayudaleyprotecciondatos.es/2016/06/06/normativa-lopd-aplicaciones-moviles/]

Per altra banda, i pel temps establert amb el pla de treball, no s'inclouran mitjans per capitalitzar l'aplicació, però si clarament es podrà monetaritzar en un futur amb publicitat, en aquest cas per part l'API de control es podrà enviar missatges publicitaris, amb les primícies o factors:

- Posició demogràfica, tant el lloc on l'usuari ha publicat més missatges, com aquells llocs temporals com vacances, centres comercials, platja, farmàcies, etc...
- Resultats segons les aprovacions positives o negatives d'altres missatges per veure gustos d'usuari.
- O com ja em vist, en les grans xarxes socials, utilització de *data learning* per ***entrenar*** al sistema, aquest ha de promocionar campanyes publicitaries en temps real segons els punts anteriors.

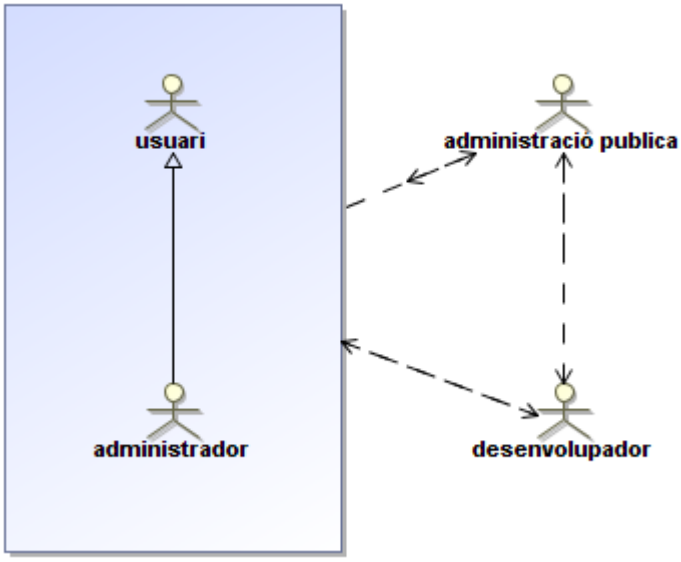
Finament, tota la possible monetarització en el futur del programari és necessari un gran volum de dades (big data), per aquest fet seria necessari un capitalització inicial, precisament per promocionar-la, per aconseguir el volum necessari d'usuaris, aproximadament mig milió.

[https://pickaso.com/2015/aumentar-retencion-usuarios-app]

5.4 Actors del sistema

Per definir el comportament del sistema, a partir del requisits anteriors, hem de presentar els actors així com la seva relació amb els casos d'ús.

Hi ha dos autors principals –usuari i administrador–, però hem de descartar les parts implicades com l'administració publica, per poder reflectir la llei **LOPD** sobre protecció de dades, i el desenvolupador.



Restricció textual: L'usuari i l'administrador s'han d'autentificar amb les seves credencials, no hi ha actors no autentificats en el sistema.

Usuari:

Usuari principal del sistema que s'ha autentificat amb les seves credencials. Aquest prèviament s'ha donat d'alta al sistema i pot accedir-hi. Podrà enviar missatges, així com veure'n d'altres i valorar-los positivament o negativament.

Administrador:

Gestor del sistema, és un usuari amb drets especials, tot i que en un primer moment no tindrà processos automatitzats, si indicara una serie de restriccions en la construcció i ús de aplicació. En tot cas, en un futur es podrà crear una extranet interna per poder gestionar els estats de bloqueig de missatges, usuaris o la eliminació d'aquests.

Administració publica:

Dins de la legalitat on es contextualitza la creació del programari, l'administració publica regula mitjançant el registre general de protecció de dades, en l'acord previst en l'article de 14 de la Lleí Orgànica 15/1999, de Protecció de Dades de Caràcter Personal (LOPD).

Desenvolupador:

Els enginyers informàtics o desenvolupadors, especialment durant les primeres etapes de vida del projecte, definiran tasques tècniques les qual podran variar des de el entorn que s'ha de crear per realitzar el projecte, així com en menor mida, les primícies pel manteniment del producte.

5.5 Model de casos d'ús

Podem extraure unes histories d'usuari com a base per a un estudi de les parts implicades en el projecte, amb les seves diferents necessitats, i per la creació dels casos d'us. Aquestes histories d'usuaris son a nivell d'exemple, poden variar durant l'anàlisi del programari.

5.5.1 Usuari:

Com a usuari vull ...

- ... publicar un missatge en el punt geogràfic actual.
- ... veure els 10 missatges de 5km a la rodona ordenats per vots positius, o ampliant el radi d'acció geogràfica fins veure 10 missatges.
- ... poder votar positivament o negativament un missatge.
- ... veure els missatges publicat anteriorment.
- ... eliminar un missatge publicat anteriorment.
- ... poder modificar el text dels missatges enviats anteriorment.

5.5.2 Administrador

Com administrador vull...

- ... que tots els usuaris estiguin autentificats.
- ... que el sistema sigui d'alta disponibilitat per a que l'usuari pugui utilitzar-la en qualsevol moment.
- ... bloquejar o desbloquejar missatges.
- ... poder bloquejar o desbloquejar usuaris.
- ... eliminar missatges.
- ... eliminar usuaris.
- ... vull que la base de dades pugi informar de registres amb la seva geolocalització.

5.5.3 Administració pública

Com administració pública vull...

- ... que s'informe als usuaris del seus drets durant el registre de les seves dades.
- ... que s'indiqui l'acotació i el propòsit de l'aplicació i l'ús de les dades privades del usuari.
- ... que l'usuari pugui en tot moment sol·licitar la eliminació de les seves dades

5.5.4 Desenvolupador

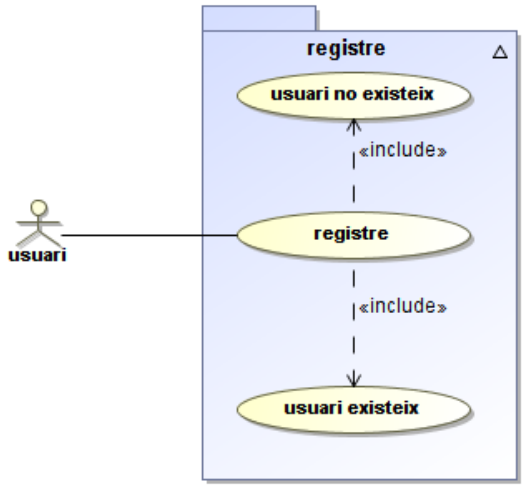
Com a desenvolupador vull...

- ... que s'aprovisione l'entorn d'una forma automatitzada.

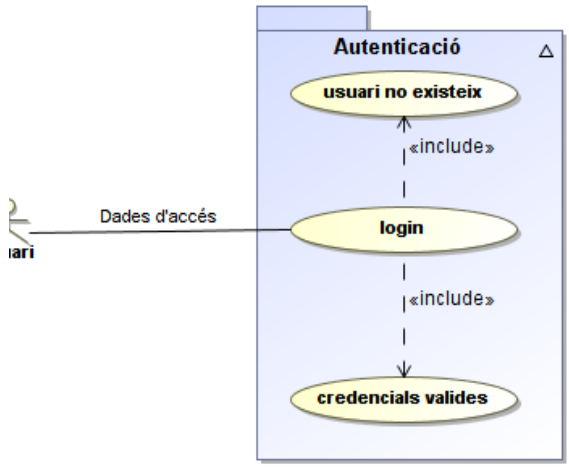
5.5.5 Casos d'ús

A partir del enunciats anteriors podem crear els diagrames principals de com interactua l'usuari amb el sistema:

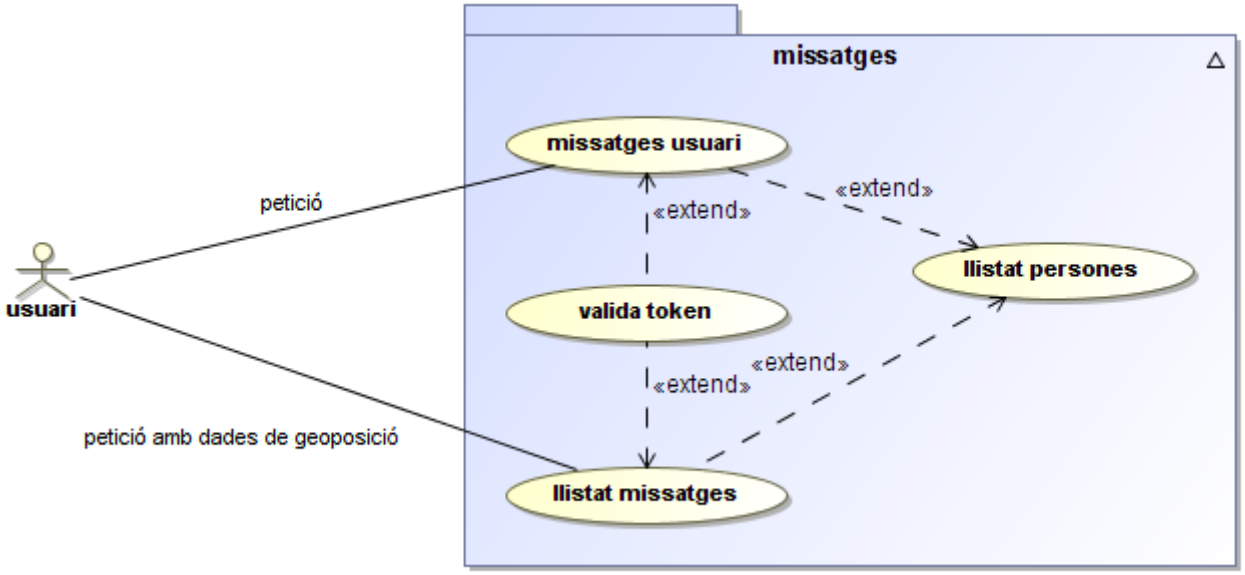
Registre:



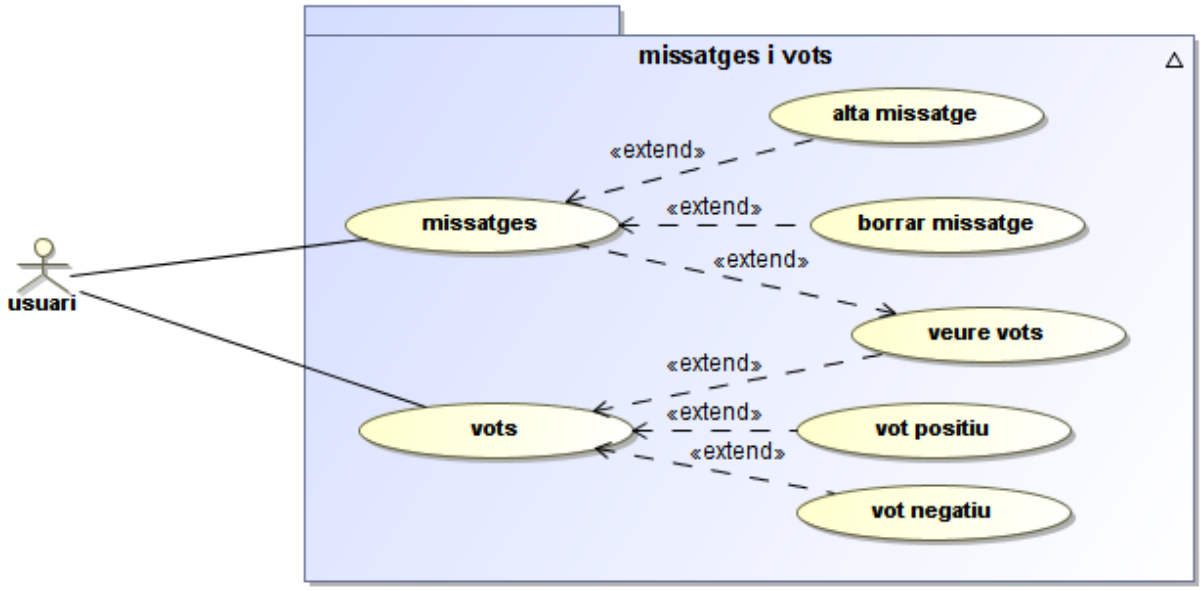
Autenticació:



Missatges:

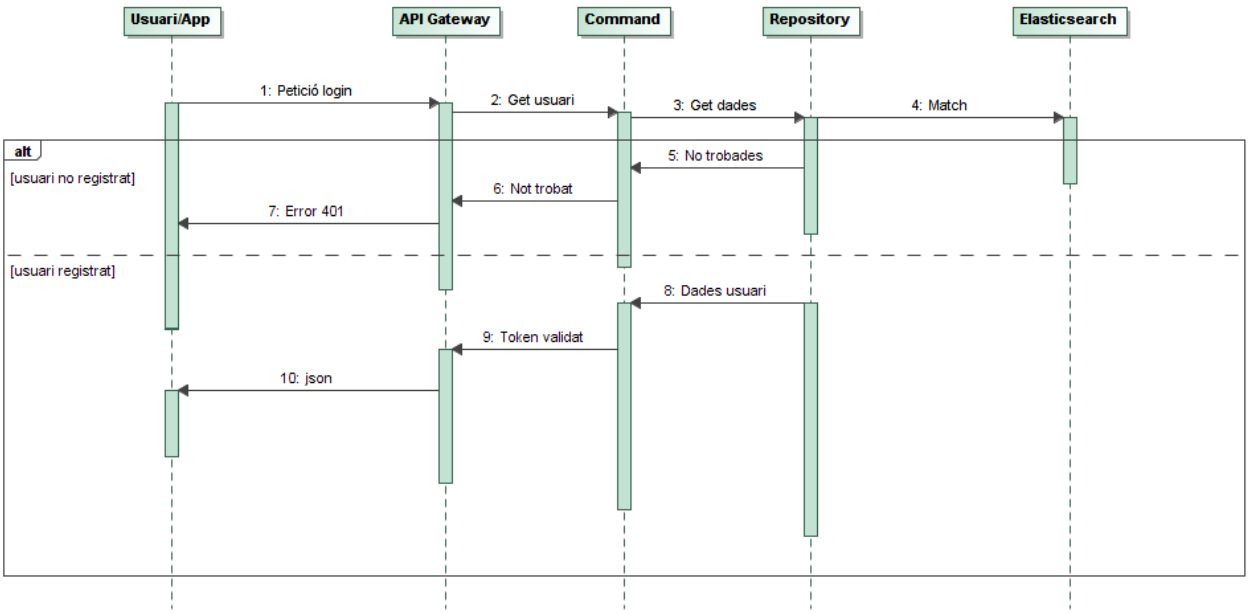


Missatges i vots:

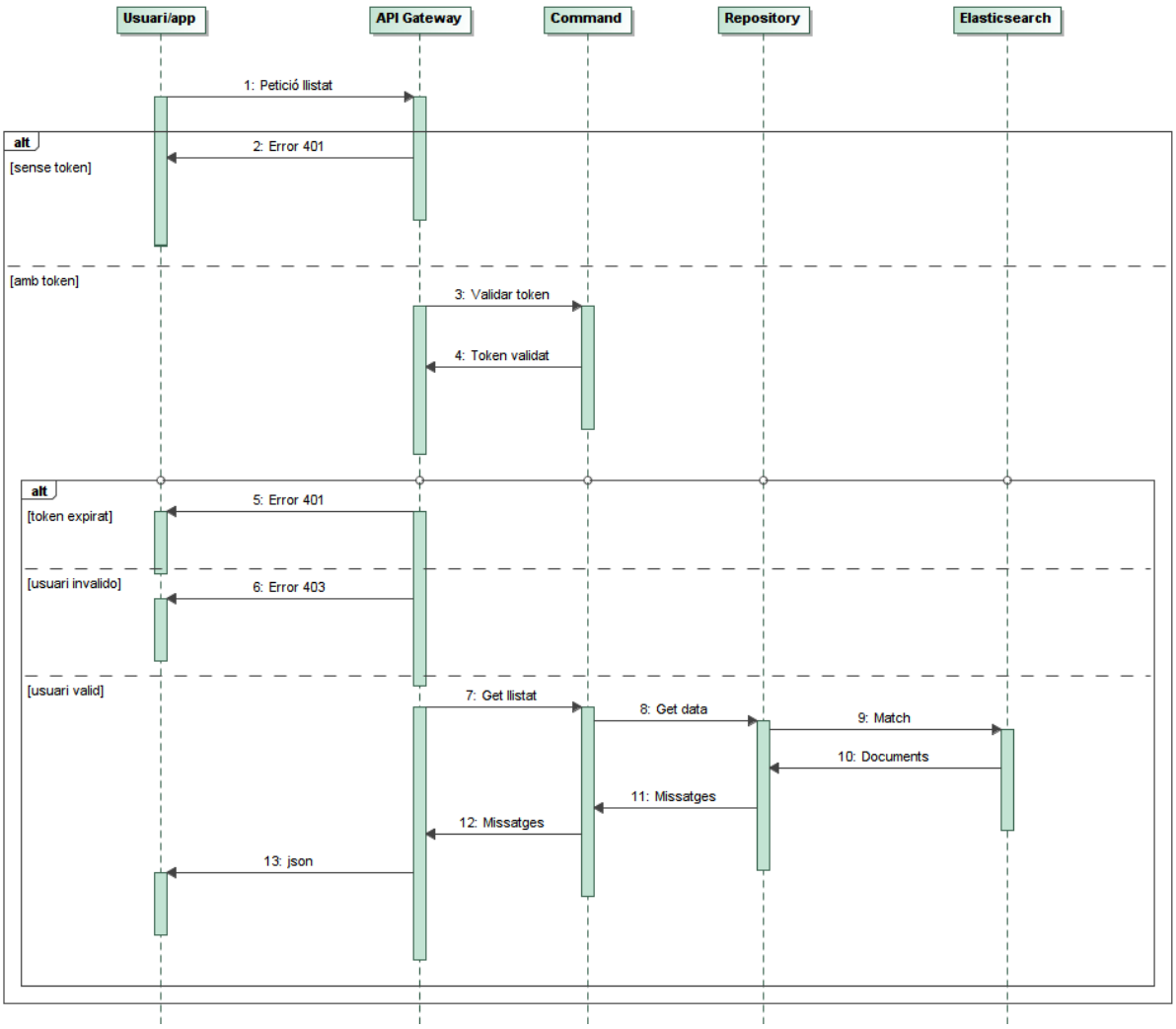


5.6 Diagrama de seqüencies

Representarem dos casos de sentencies principals de l'aplicació per ser complexos i representatius, de la gestió entre usuari amb l'aplicació frontal i la seva iteració amb la API més base de dades. El cas que l'autentificació d'usuari, abans s'ha registrat, obtindrà un token per futures peticions:



I finalment, l'usuari amb un token valid intenta accedeix per veure el llistat de missatges de la seva geoposició:



5.7 Fitxes de casos d’ús

Crearem fitxes de casos d’ús del casos mes representatius i, en tot cas, mes complexos del programari.

Cas d'ús	Registre d’usuari i acceptació de la política de privacitat.
Actor principal	usuari
Àmbit	Aplicació de missatges geolocalitzats per dispositius mòbils.
Nivell d'objectiu	usuari
Stakeholders i interessos	
Usuari d’aplicacions mòbils: L’usuari vol utilitzar l’aplicació per primer cop per veure el missatges publicats en las seva posició actual. Administració publica: vol que es conservin, segons la normativa, les polítiques de privacitat de les dades proporcionades pels usuaris.	
Precondició	Haver instal·lat la aplicació en el dispositius mòbil.
Garanties mínimes	-
Garanties en cas d'èxit	El sistema mostrara el 10 missatges mes valorats de la posició actual del usuari, amb data de creació mes recent, i obtindrà un token per utilitzar temporalment l’aplicació fins que caduqui.
Escenari principal d'èxit	
1) L’usuari accedeix a l'aplicació per primer cop. 2) L’aplicació mostra la vista d’autentificació amb els camps de nom d’usuari, clau de pas, boto per enviar les dades, i un accés per crear un compte. 3) L’usuari selecciona «create account» 4) L’aplicació mostra el formulari de registres, on apareixen els camps: nom, data de naixement, nom d’usuari, clau de pas, selector per política de privacitat i el boto per enviar les dades. 5) L’usuari indica les dades correctament i seleccionar el boto «enviar» 6) El programari mostra un missatge de registre complet, i mostra els 10 missatges ordenats per posició, vots positius i data de creació més recent. I es mostra la metàfora de publicació de missatge,	
Extensions	
4) L'usuari no introdueix algun de camps i seleccionar enviar. 4.1) El sistema mostra l’error informant quin camp està incorrecte, el sistema torna a al punt anterior. 5) L’usuari selecciona sobre la frase «privacy policy» 5.1)El sistema mostra un finestra flotant amb la informació de la política de privacitat. 5.2) L’usuari tanca la finestra selecciona el boto «close», el sistema torna al punt	

anterior.

Cas d'ús	Publicar un missatge amb les dades de geolocalització
Actor principal	usuari
Àmbit	Aplicació de missatges geolocalitzats per dispositius mòbils
Nivell d'objectiu	usuari
Stakeholders i interessos	
Usuari d'aplicacions mòbils: L'usuari vol publicar un missatge amb la seva posició actual. Administrador: Vol que l'usuari pugui enviar un missatge en qualsevol moment.	
Precondició	Haver realitzar el registre amb el programari.
Garanties mínimes	-
Garanties en cas d'èxit	El sistema actualitzara el llistat de missatge publicats per l'usuari.
Escenari principal d'èxit	
1) L'usuari accedeix a la aplicació. 2) L'aplicació mostra la vista d'autentificació amb els camps de nom d'usuari, clau de pas, boto per enviar les dades, i un accés per crear un compte. 3) L'usuari introdueix correctament les dades d'usuari. 4) El sistema mostra els 10 missatges ordenats per posició, vots positius i data de creació més recent. I es mostra la metàfora de publicació de missatge, a més de les dades del usuari a la part superior de l'aplicació. 5) L'usuari selecciona la icona de publicació de missatge. 6) L'aplicació mostra una vista on podem veure un àrea de escriptura, el teclat en la part inferior de la pantalla. 7) L'usuari selecciona l'àrea d'escriptura i introdueix el text del missatge. 8) Al escriure text el sistema torna a mostra la metàfora d'enviar missatge. 9) L'usuari seleccionar la icona enviar missatge. 10) El sistema actualitzara el llistat de missatge publicats per l'usuari	
Extensions	
3) L'usuari no introdueix algun de camps i seleccionar enviar. 3.1) El sistema mostra l'error informant quin camp està incorrecte, el sistema torna a al punt anterior. 7) L'usuari selecciona el boto del dispositiu mòbil «tornar enrere» 7.1) El sistema tornar al punt 4) 10) El sistema no te connectivitat a internet. 11) El sistema informa que s'ha produït un error i no te accés a internet.	

Cas d'ús	Votar positivament un missatge del llistat
Actor principal	usuari
Àmbit	Aplicació de missatges geolocalitzats per dispositius mòbils
Nivell d'objectiu	usuari
Stakeholders i interessos	
Usuari d'aplicacions mòbils: L'usuari lleig un missatge de la geoposició actual i el vota positivament. Administrador: Vol que l'usuari pugui votar un missatge en qualsevol moment.	
Precondició	Haver autentificat anteriorment o tenir el token valid.
Garanties mínimes	-
Garanties en cas d'èxit	El sistema actualitzara els vots del missatge votat.
Escenari principal d'èxit	
1) L'usuari accedeix a la aplicació. 2) El programari mostra els 10 missatges ordenats per posició, vots positius i data de creació més recent. I es mostra la metàfora de publicació de missatge, a més de les dades del usuari a la part superior de la pantalla. 3) L'usuari lleig els missatges, un dels qual el selecciona. 4) El sistema mostra sobre el missatge 2 metàfores de vot positiu o negatiu. 5) L'usuari selecciona la icona de vot positiu. 6) El sistema mostra un missatge de vot correcte i torna al punt 2.	
Extensions	
5) L'usuari selecciona la icona de vot negatiu, continua al punt 6) 6) El sistema no te connectivitat a internet 6.1) El sistema mostra un missatge d'error on indica que no te accés a internet, es torna al punt 2).	

Cas d'ús	Eliminar un missatge publicar anteriorment
Actor principal	usuari
Àmbit	Aplicació de missatges geolocalitzats per dispositius mòbils
Nivell d'objectiu	usuari
Stakeholders i interessos	
Usuari d'aplicacions mòbils: L'usuari vol eliminar un missatge publicat anteriorment al sistema. Administrador: Vol que l'usuari pugui borra un missatge en qualsevol moment.	
Precondició	Haver autentificat anteriorment o tenir el token valid, i haver publicar un missatge anteriorment.
Garanties mínimes	-
Garanties en cas d'èxit	El sistema actualitzara els llistat de publicacions del usuari.
Escenari principal d'èxit	
1) L'usuari accedeix a la aplicació. 2) El sistema mostra els 10 missatges ordenats per posició, vots positius i data de creació més recent. I es mostra la metàfora de publicació de missatge, a més de les dades del usuari a la part superior de la pantalla. 3) L'usuari selecciona el seu nom d'usuari de la part superior de la pantalla. 4) El sistema canvia el llistat de missatges pels missatges del usuari publicats anteriorment, ordenats per data de creació mes recent, a la dreta de cada missatge ara apareix una metàfora que indica que es pot eliminar el missatge. 5) L'usuari selecciona la icona d'eliminació d'un missatge. 6) El sistema mostra un missatge que s'ha eliminar correctament i desapareix, el programari tornar al punt 4).	
Extensions	
5) L'usuari selecciona el boto del dispositiu mòbil per tornar enrere. 6) El sistema torna al punt 2) 6) El sistema no te connectivitat a internet 6.1) El sistema mostra un missatge d'error on indica que no te accés a internet, es torna al punt 4).	

6. Disseny

En aquesta fase em de plantejar els conceptes, usabilitat i expectatives de disseny per la implementació del programari, un disseny centrar en l'usuari.

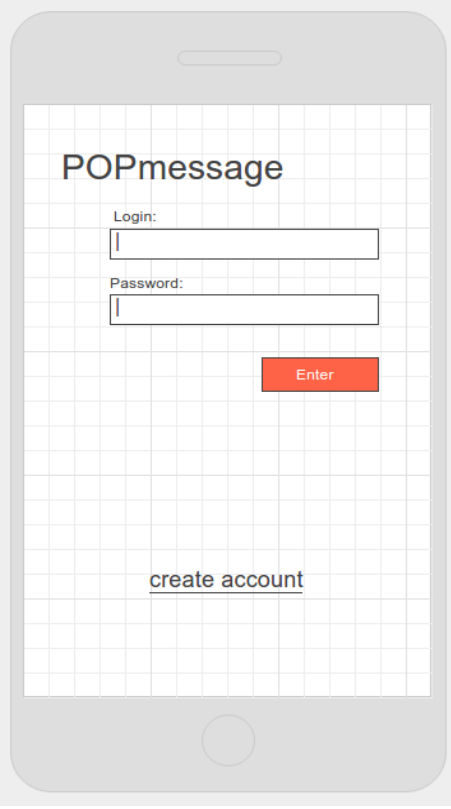
A més, presentar els diagrames de classes tant de l'aplicació Android® com la de API d'enllaç, així com els «*endpoints*» d'aquesta. Pel que fa la persistència o base de dades és basara en la primícia d'alta disponibilitat i distribució, sacrificant l'atomicitat de les dades i la seva relació d'integritat.

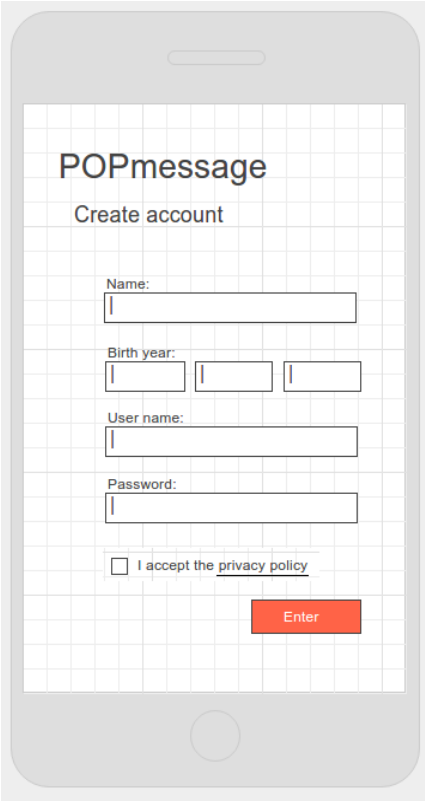
Finalment, un diagrama a nivell global per veure l'arquitectura de l'aplicació, on es veuran els components importants.

6.1 Models de pantalles (prototipus)

6.1.1 Interfícies

La interacció «persona-ordinador» en la primera fase de vida s'espera una aplicació limitada, posteriorment es podrà ampliar, depenen d'aquest fet de les necessitats dels usuaris o evolutiu del mercat. El primer cop que s'utilitza la aplicació es necessari un registre, en tot cas el primer pas es sol·licitar les dades d'accés amb opció a crear un compte d'usuari nou:

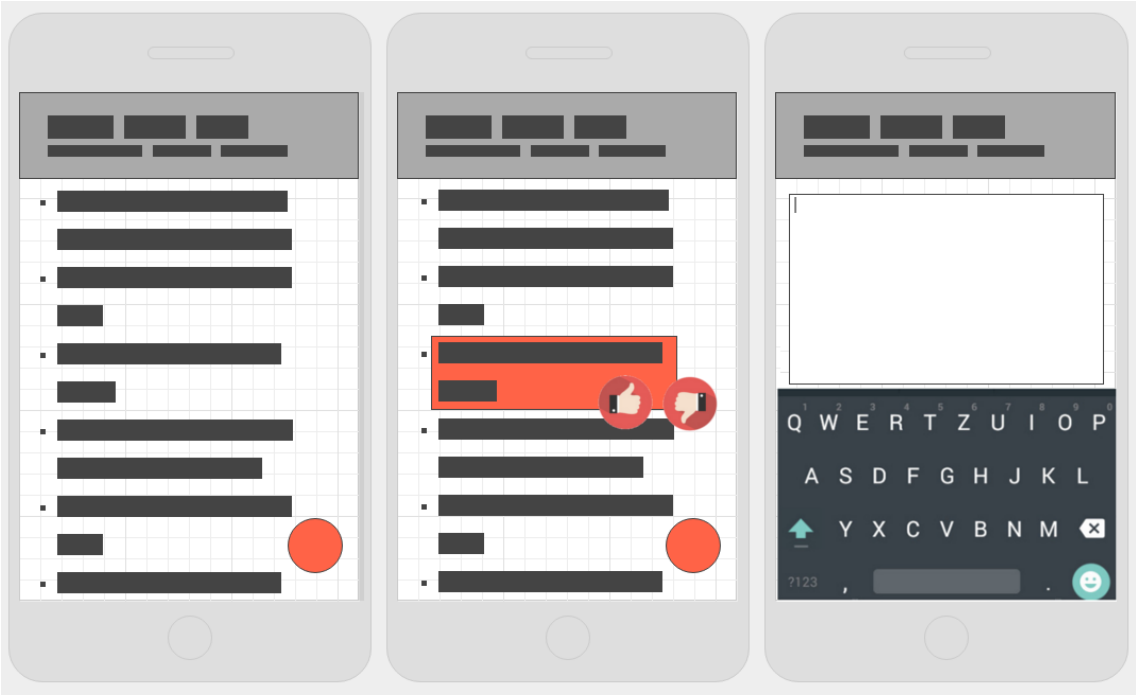




El 80% de la funcionalitat en l'ús de la aplicació està visible des de el primer moment, el llistat de missatges i la icona (metàfora) de la part inferior dreta per enviar un missatge s'han de mostrar en un primer moment, si l'usuari està autentificat.

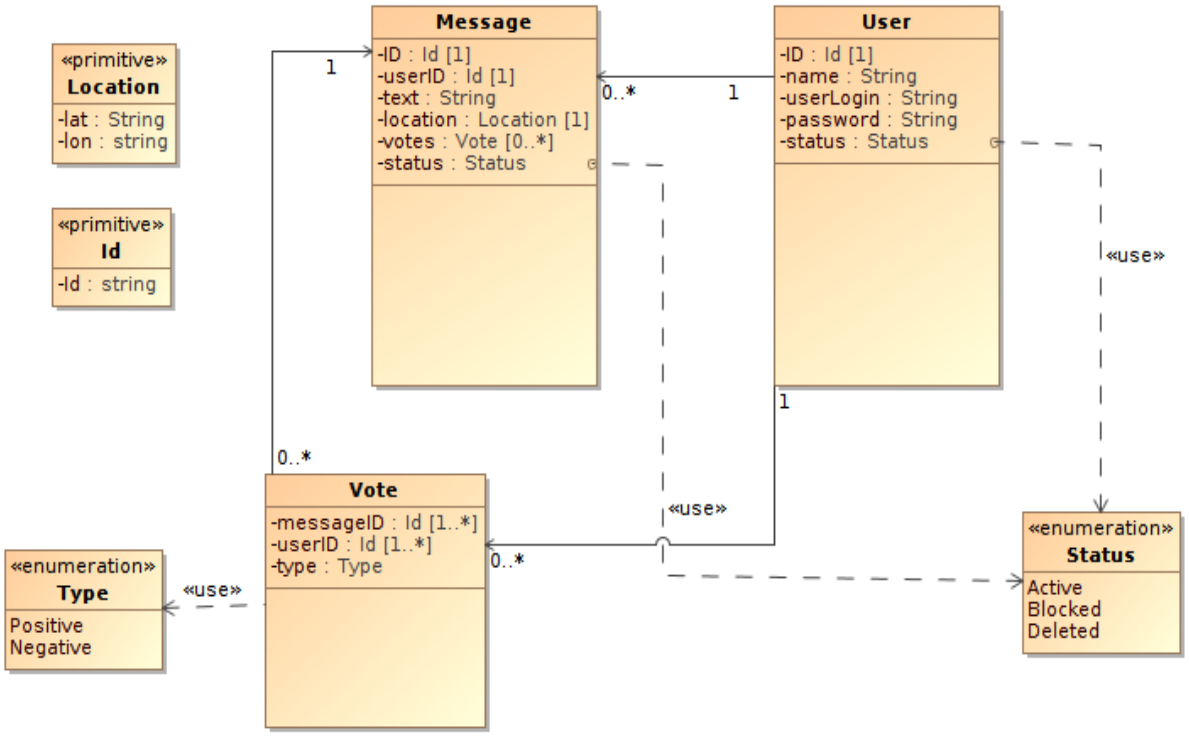
En tot cas, la icona o metàfora per enviar un missatge ha de ser visible en tot moment, menys quan s'està escrivint o enviar el contingut.

Per realitzar la votació d'un contingut s'ha de seleccionar i prémer posteriorment el tipus de vot (positiu o negatiu) que estaran representades per una metàfora.



6.2 Diagrama de classes principals

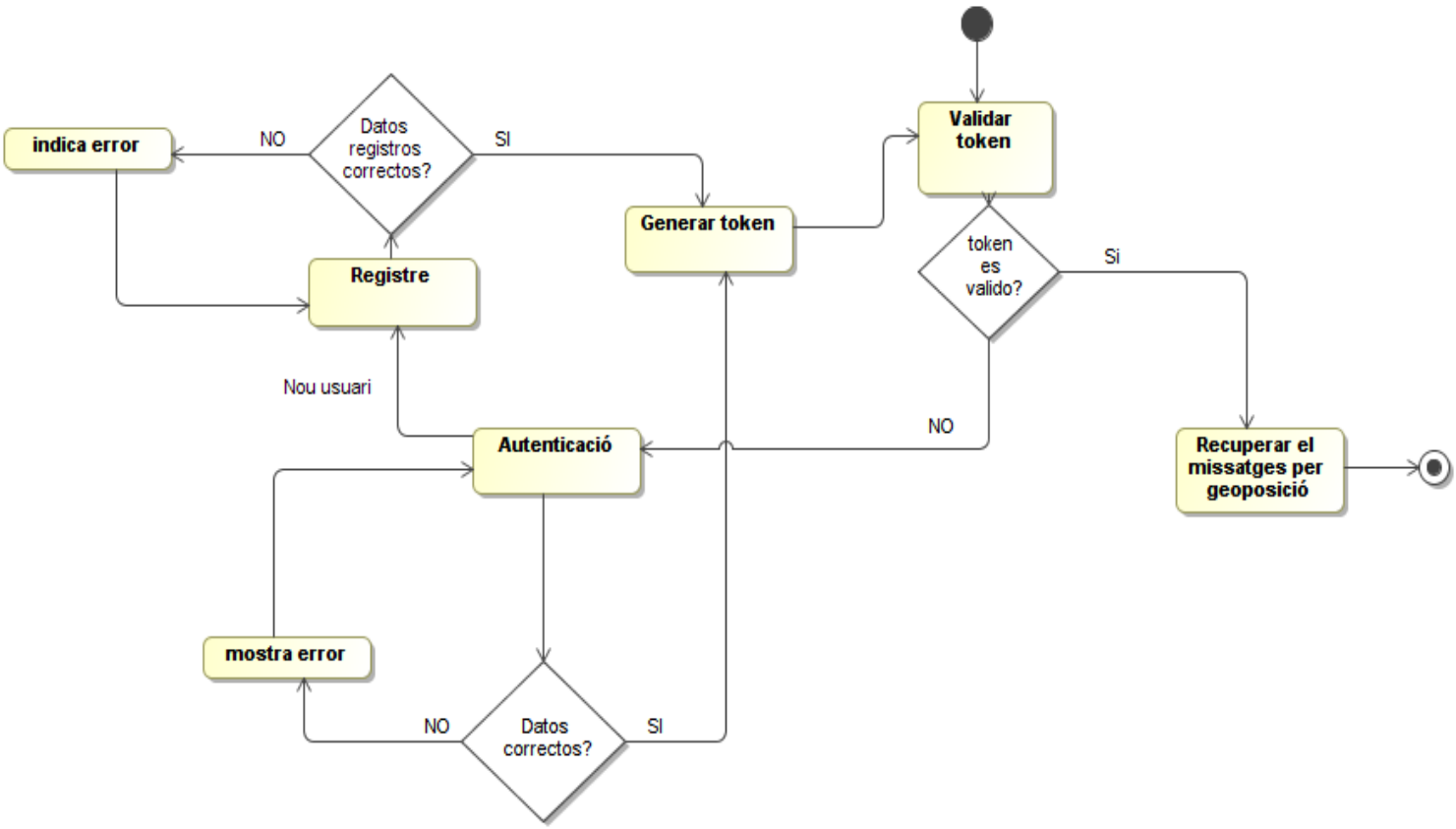
Els diagrames UML de les **entitats del domini,** en el paradigma de orientació objectes, tenim les classes que modelen la informació i les seves relacions. Em de destacar l'abstracció dels missatges, vots i usuaris, amb les seves relacions d'integritat.



6.3 Diagrama d’estats

La realització del diagrama d’estats ha d’explicar els comportament del sistema, o en tot cas, la part més representativa.

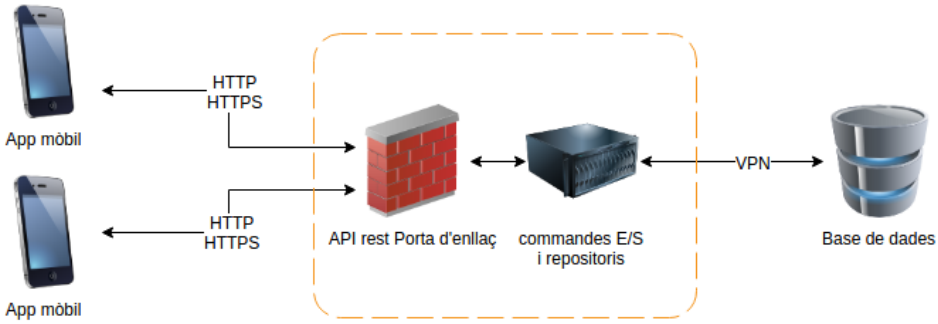
Per aquest fet, la representació d'utilització del *token* en el procés d’autentificació i registre ens explicara el flux principal d’iniciació de l’aplicació:



A partir del punt final d’aquest diagrama (recuperar missatges per geoposició) s’iniciaria la resta d’interaccions de l’usuari amb el programari: votar o publicar un missatges, veure i eliminar missatges publicats anteriorment,...

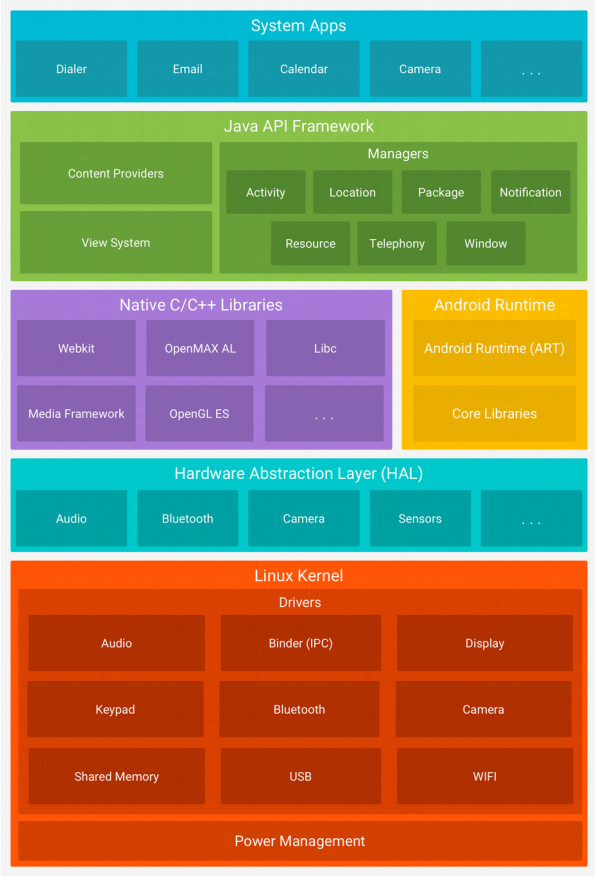
6.4 Diagrama d'arquitectura

Com a model conceptual podem veure el components de l'arquitectura amb el següent diagrama, on estan destacades, en 3 capes: la part frontal, la porta d'enllaç i la base de dades de persistència.



6.4.1 Frontal – App Android®

Android es un sistema operatiu basat en Linux per a dispositius mòbils creat per Google. L'arquitectura Android es està composta pels següents components:



En la base de la plataforma està el **kernel de linux**, per exemple en temps d'execució es base en el kernel per a funcionalitats subjacents, com la generació de subprocessos i l'administració de memòria a baix nivell.

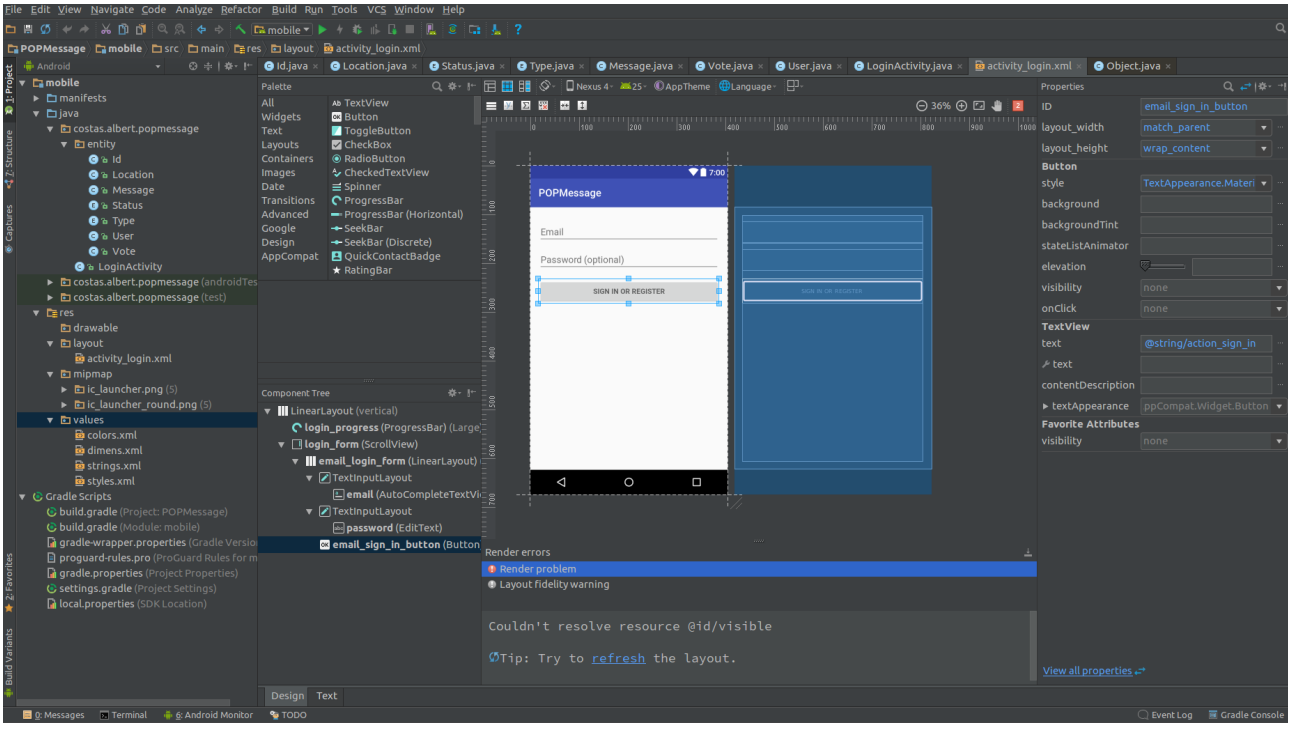
La **capa d'abstracció del hadware (HAL)** ens ofereix un interfície estàndard per usar les capacitats del maquinari del dispositiu amb al marc de treball Java API, ens aquest cas per obtenir la geoposició que ens interessa per al nostre projecte.

En aquest sentit al *Framework* de Java API ens ofereix el recursos per a treballar en l'entorn de desenvolupar:

- sistema de vista
- administració de recursos
- administració de notificacions
- administració d'activitat
- proveïdor de contingut

Android Studio®

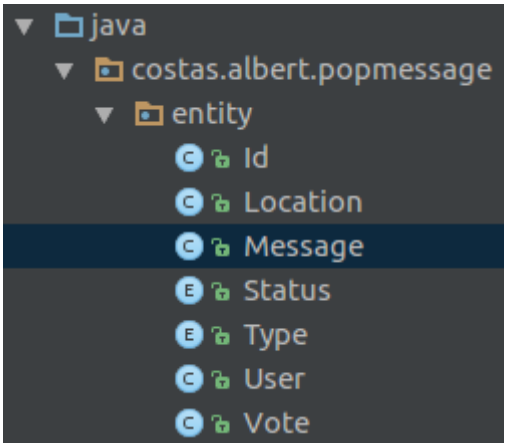
Inicialitzem el projecte, amb les carpetes de les classes java (paquets d'entitats, activitats,...), el recursos amb les diferents i el manifest:



Podem veure el projecte que ja tenim al repositori de codi (github.com): https://github.com/acostasg/POPmessage

Per obtenir el codi:

```
$ git clone git@github.com:acostasg/POPmessage.git
```



Per fer la conversió dels objectes json a les entitats que en proveirà la API poden usar eines com:

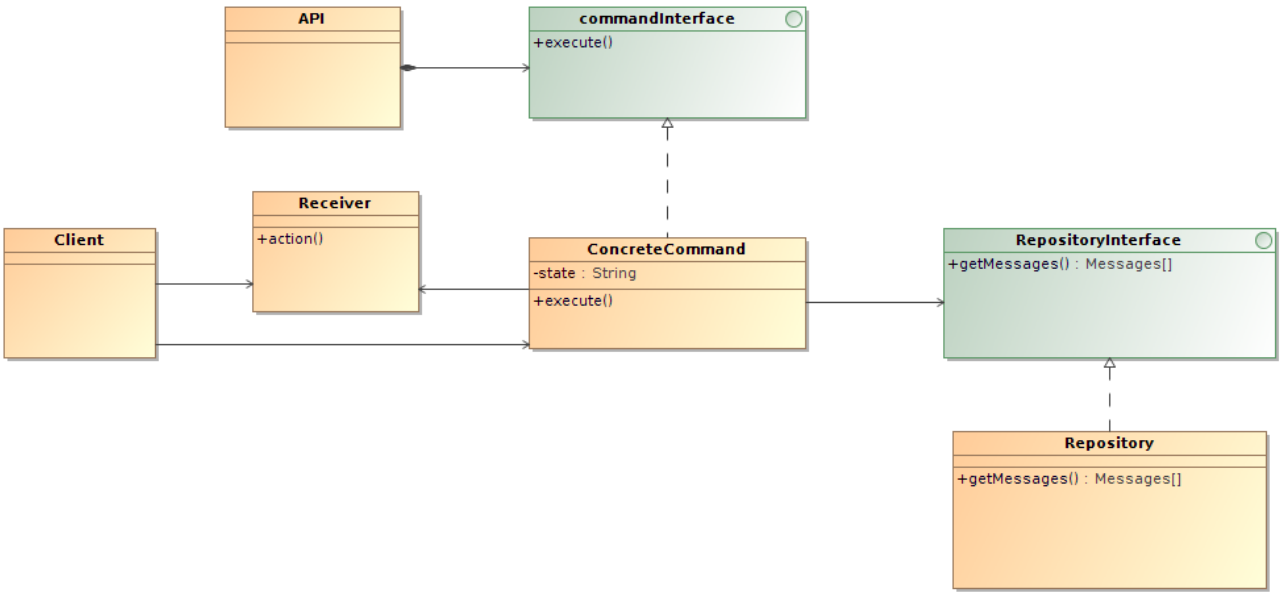
http://www.jsonschema2pojo.org/

Amb la instal·lació de la llibreria *com.fasterxml.jackson* que en permet decorar les entitats per una conversió amb els format JSON.

Aquestes entitats son del domini de l'aplicació, i estan representades l'àmbit del programari, basades en el punt 6.2 de la memòria.

6.4.2 API Porta d'enllaç

Els patrons a utilitzar el podem representar en el següent gràfic **UML**. A partir del punts d'entrada de la porta d'enllaç (API) saben que cada «endpoint» de la api representara una **comanda** que executa un procés, i aquest podrà **consultar o persistir delegant a un repositori**:



Degut a l'arquitectura de l'aplicació, en una primera versió no hi haurà suport mult-idioma.

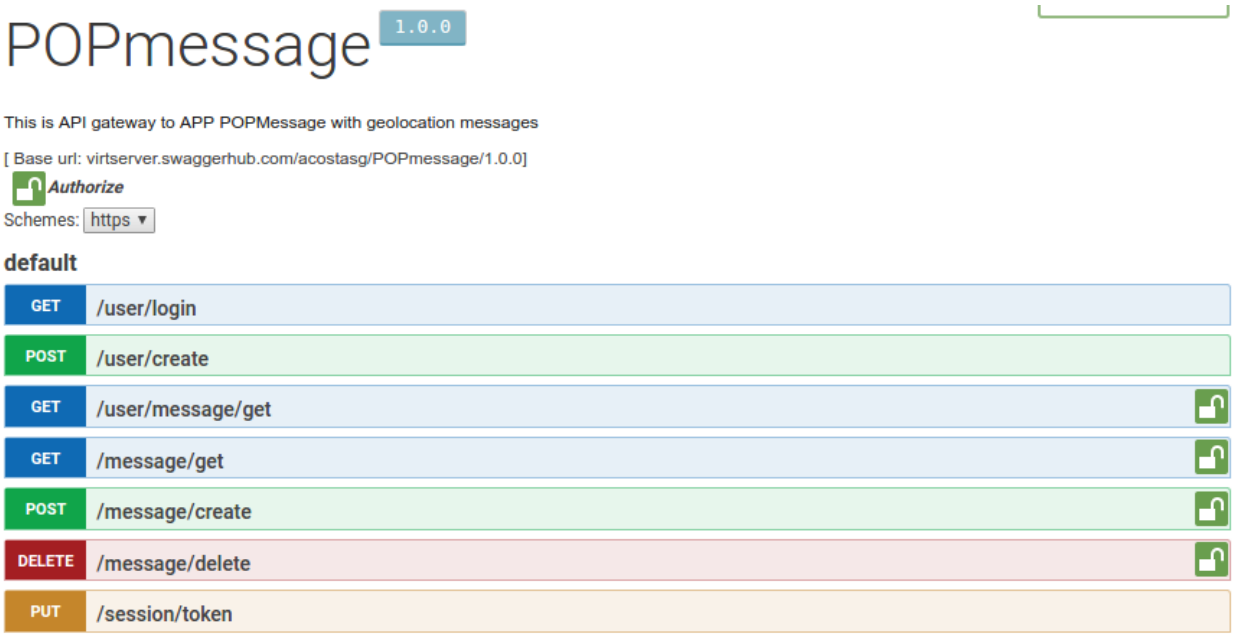
Pel que fa ens punts d'entrada ho documentarem amb *Swagger*, que ens proveeix d'un marc de treball per descriure, produir, consumir i visualitzar els serveis web RESTful, amb un llenguatge agnòstic.

Como API s'ha de complir la següent tabla:

HTTP Verb	CRUD	Entire Collection (e.g. /customers)	Specific Item (e.g. /customers/{id})
POST	Create	201 (Created), 'Location' header with link to /customers/{id} containing new ID.	404 (Not Found), 409 (Conflict) if resource already exists..
GET	Read	200 (OK), list of customers. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single customer. 404 (Not Found), if ID not found or invalid.
PUT	Update/Replace	404 (Not Found), unless you want to update/replace every resource in the entire collection.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
PATCH	Update/Modify	404 (Not Found), unless you want to modify the collection itself.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
DELETE	Delete	404 (Not Found), unless you want to delete the whole collection—not often desirable.	200 (OK). 404 (Not Found), if ID not found or invalid.

[http://www.restapitutorial.com/lessons/httpmethods.html]

API «endpoints»



GET /user/login

PARAMETERS — Try it out

NAME	DESCRIPTION
userName * string (query)	User name to login user
password * string (query)	Password for login user encrypted

RESPONSES

Response content type: application/json

CODE	DESCRIPTION
200	OK
400	Invalid username and password
404	user not found

POST /user/create

PARAMETERS — Try it out

NAME	DESCRIPTION
name * string (formData)	Real name user
dateOfBirth * string (formData)	Date of birth user
userName * string (formData)	User name to login user
password * string (formData)	Password for login user encrypted
privacyPolicy * boolean (formData)	Accept the privacy policy

RESPONSES

Response content type: application/json

CODE	DESCRIPTION
200	OK
400	Invalid username and passwordcombination
409	Username already exist

GET /message/get

PARAMETERS Try it out

NAME	DESCRIPTION
lat * string (query)	Latitude
lon * string (query)	Longitude
token * string (query)	Token session

RESPONSES

Response content type: application/json

CODE	DESCRIPTION
200	OK
400	Invalid latitude or longitude
404	Not found messages

GET /user/message/get

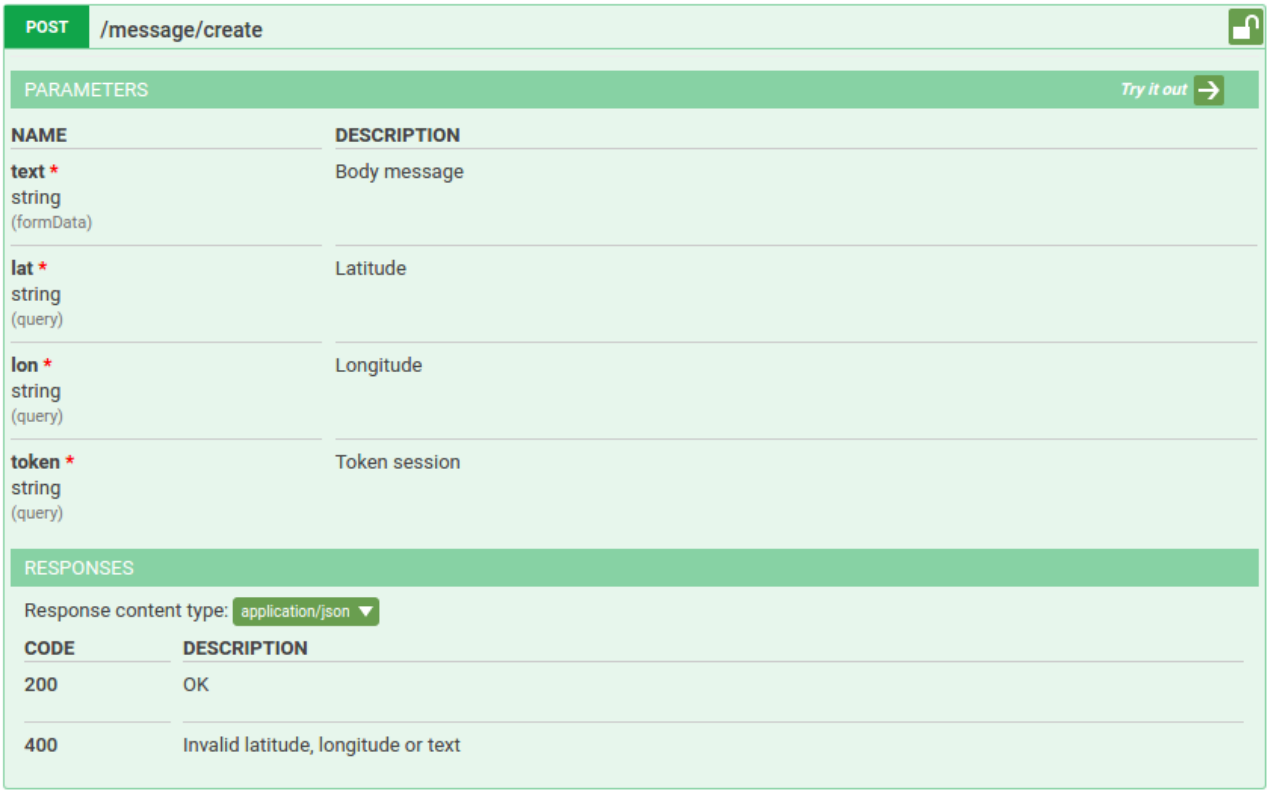
PARAMETERS Try it out

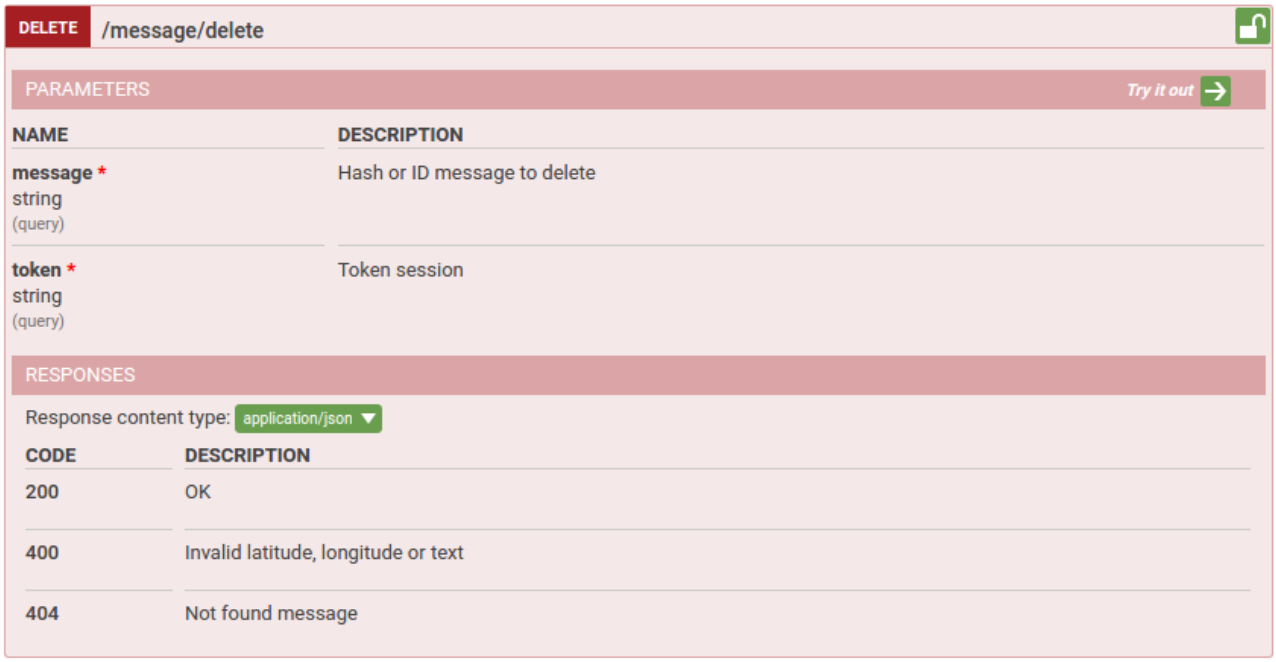
NAME	DESCRIPTION
token * string (query)	Token session

RESPONSES

Response content type: application/json

CODE	DESCRIPTION
200	OK
400	Invalid latitude or longitude
404	Not found messages


POST
/message/create
PARAMETERS
Try it out
NAME
DESCRIPTION
text *
string
(formData)
Body message
lat *
string
(query)
Latitude
lon *
string
(query)
Longitude
token *
string
(query)
Token session
RESPONSES
Response content type: application/json
CODE
DESCRIPTION
200
OK
400
Invalid latitude, longitude or text


DELETE
/message/delete
PARAMETERS
Try it out
NAME
DESCRIPTION
message *
string
(query)
Hash or ID message to delete
token *
string
(query)
Token session
RESPONSES
Response content type: application/json
CODE
DESCRIPTION
200
OK
400
Invalid latitude, longitude or text
404
Not found message

Més informació: https://app.swaggerhub.com/api/acostasg/POPmessage/1.0.0

Important destacar que hi ha tres punts de entrada que no és necessari token valid d'usuari (registre, login i validació de token), però si en qualsevol «endpoint» es necessari en la capçalera de la petició (http request) la API key (clau d'aplicació) que correspon a la APP d'Android POPmessage.

Ens basarem per la gestió del token i la la clau d'API per l'aplicació amb les bones practiques per REST *token* amb autenticació JAX-RS i Jersye. [http://stackoverflow.com/questions/26777083/best-practice-for-rest-token-based-authentication-with-jax-rs-and-jersey]

Destaquem de la resposta de la web http://stackoverflow.com com a resum del flux d'autenticació amb *token*:

> ***How token-based authentication works***
>
> In a token-based authentication, the client exchanges *hard credentials* (such as username and password) for a piece of data called *token*. Instead of sending the hard credentials in every request, the client will send the token to the server to perform authentication and authorization.
> In a few words, an authentication scheme based on tokens follow these steps:
>
> 1.The client sends their credentials (username and password) to the server.
> 2.The server authenticates the credentials and generates a token.
> 3.The server stores the previously generated token in some storage along with the user identifier and an expiration date.
> 4.The server sends the generated token to the client.
> 5.In every request, the client sends the token to the server.
> 6.The server, in each request, extracts the token from the incoming request. With the token, the server looks up the user details to perform authentication and authorization.
>
> 1.If the token is valid, the server accepts the request.
> 2.If the token is invalid, the server refuses the request.
>
> 7.The server can provide an endpoint to refresh tokens.
>
> Cássio Mazzochi Molin

Més informació sobre la codificació dels token https://jwt.io/.

6.4.3 Persistència

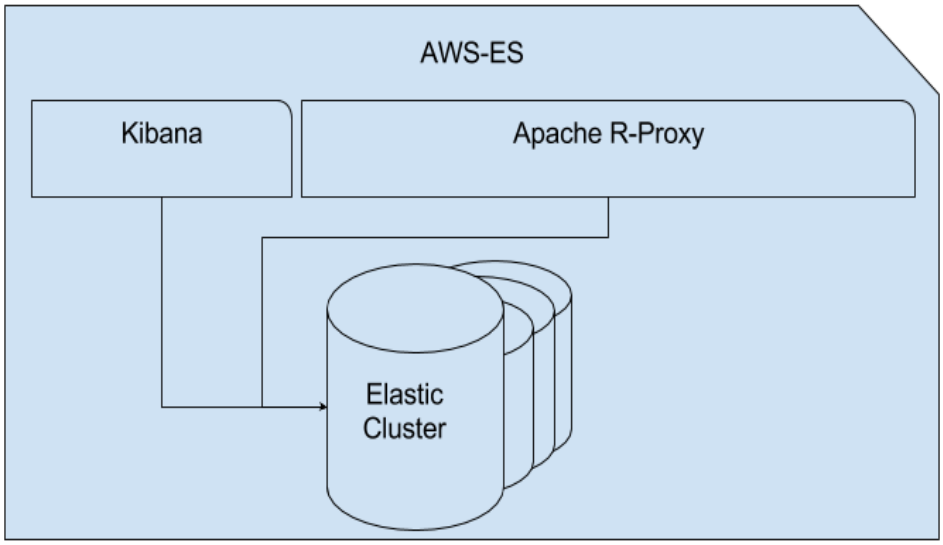
La representació de les entitats del domini (punt 6.2) en la seva translació en la base de dades o capa de persistència es prioritzara la utilització de dades **des-normalitzades** per a cerques rapides, per tant s'intentara garantir la disponibilitat d'aquesta informació al fet que siguin atòmiques.

Degut aquest fet, utilitzarem bade de dades **noSql** (documentals, grafs, etc...), per exemple hi haurà documents que representaran missatges on es repetira el nom del usuari o numero de vots, pel fet de disposar d'una forma rapida d'aquesta informació en les consultes.

ElasticSerch®

Per aquesta motiu s'ha seleccionat *ElasticSearch®* , una base de dades documental, distribuïda, escalable i d'alta disponibilitat. És una potent eina que ens permet indexar una gran volum de dades i posteriorment fer consultes sobre aquestes suportant cerques aproximades i agregats sobre aquestes. Un ús pot ser fer consultes de text complet, en estar les dades indexades, els resultats s'obtenen de forma molt ràpida.

Aquesta utilitza una API *RESTfull* sobre consultes protocol HTTP, indexació i administració en diferents llenguatges. La indexació dels camps dels documents es realitza en JSON degut que no te un esquema rígid.



Estructura de dades:

Degut a que no disposem d'un esquema rígid l'objectiu es representarem les entitats de domini. Elasticsearch® en permet consultes sobre un o varies índex, inclús por tipis de document independent del index on es persisteix.

Per aquest fet, s'han de definir els «index» així com relació amb les entitats, aquests últims ens defineix l'esquema no rígid dels documents.

```
PUT /message_index
{
  "message_index": {
    "mappings": {
      "message": {
        "properties": {
          "ID": {
            "type": "string"
          },
          "user": {
            "type": "nested",
            "properties": {
              "ID": {
                "type": "string"
              },
              "name": {
                "type": "string"
              }
            }
          },
          "text": {
            "type": "string"
          },
          "location": [
            {
              "lat": "string",
              "lon": "string"
            }
          ],
          "voteCollection": [
            {
              "vote": {
                "type": "nested",
                "properties": {
                  "userID": {
                    "type": "string"
                  },
                  "nameUser": {
                    "type": "string"
                  },
                  "type": {
                    "type": "short"
                  }
                }
              }
            }
          ],
          "status": {
            "type": "short"
          },
          "crateAt": {
            "type": "date"
          }
        }
      }
    }
  }
}
```

[https://github.com/acostasg/Docker/blob/master/ElasticSearchCluster/schema/messages_index.json]

Les dades estan des-normalitzades, es a dir, no son atòmiques en aquest tipus de base de dades, com podem veure en el mateix index de missatges, tenim objectes duplicats d'usuari, no pas dels vots on aquests en estar agregats al missatges, tot i que, si és repeteix el nom de l'usuari en aquests.

Aquest fet ens permetrà respondre a peticions concurrents d'una forma molt més rapida que en una base de dades transaccional.

```
PUT /user_index
{
  "user_index": {
    "mappings": {
      "message": {
        "properties": {
          "ID": {
            "type": "string"
          },
          "name": {
            "type": "string"
          },
          "userLogin": {
            "type": "string"
          },
          "password": {
            "type": "string"
          },
          "status": {
            "type": "short"
          },
          "crateAt": {
            "type": "date"
          }
        }
      }
    }
  }
}
```

[https://github.com/acostasg/Docker/blob/master/ElasticSearchCluster/schema/user_index.json]

```
PUT /token_index
{
  "token_index": {
    "mappings": {
      "token": {
        "properties": {
        "userID": {
            "type": "string"
          },
         "hash": {
            "type": "string"
          },
         "TTL": {
            "type": "integer"
          },
          "crateAt": {
             "type": "date"
          }
        }
      }
    }
  }
```

}
[https://github.com/acostasg/Docker/blob/master/ElasticSearchCluster/schema/token_index.json]

Més informació:

- http://www.davinci-ti.es/introduccion-a-elasticsearch-y-como-instalarlo/
- https://es.wikipedia.org/wiki/Denormalizaci%C3%B3n_(base_de_datos)
- https://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf
- https://www.elastic.co/guide/en/elasticsearch/reference/5.3/index.html

Seguretat:

Sols els repositoris de l'API podran accedir a la base de dades, aquesta estarà en entorn productius en una xarxa virtual privada (VPN) amb accés restringit.

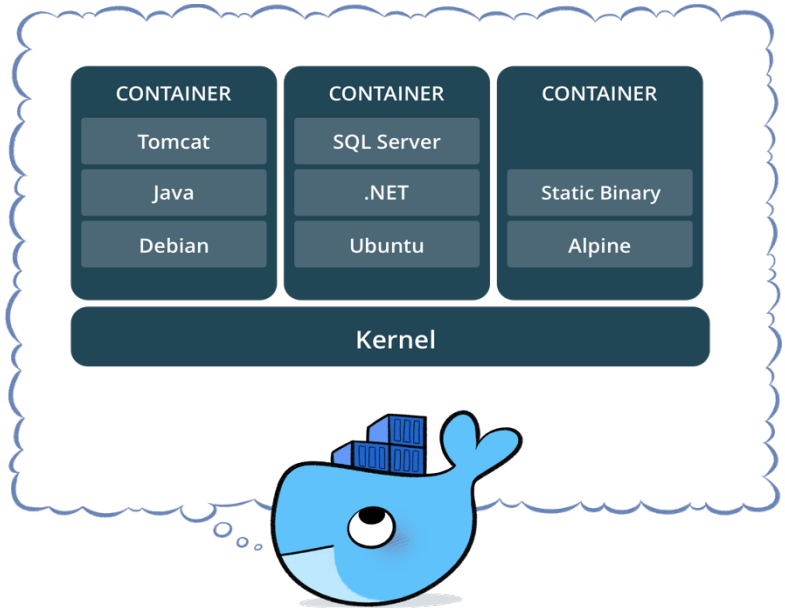
A mes s'utilitza un xarxa de *docker* on els containers o serveis que contenen sols es comuniquen en aquesta àrea privada, on sols el port 8080 es d'accés public, i requerix d'una APPKey (clau d'aplicació) em la capçalera de la petició HTTP/HTTPS per autentificar les aplicacions que hi poden accedir.

A més, a més nivell en la xarxa privada hi ha un servei *Apache* que realitza la funció de proxy invers per accedir a Elsticsearch.

Finalment, per motiu de no disposar d'un certificar d'una entitat certificadora no s'ha pogut implementat que la comunicació entre APP i la API sigui per **SSL**, en tot cas, tota la informació està xifrada per clau simètrica, no es totalment segur, però almenys les dades no s'estan enviant en pla en el protocol de comunicació HTTP.

En tot cas, no es complicat implementar la comunicació **SSL** si es disposa d'una certificació valida.

8. Entorn de desenvolupament



[https://www.docker.com/what-container]

Pel que fa l'entorn de desenvolupament hem de diferenciar la part d'experiència d'usuari, on és realitzara, com hem vist al punt 6.4.1, amb l'Android Studio®. Aquest ens proporciona un IDE (entorn integrat de desenvolupament) complet, amb un emulador de SO Android® per veure l'execució de l'aplicació.

Pel que fa la part de «backend», aquesta utilitzara Docker per la creació de container aïllats d'execució i amb la aplicació Docker-composer per aprovisionar el sistema. Hi hauran dos contenidors completament aïllats del sistema, on accedirem al port que s'hi assignara, recordem que **un contenidor sols ha de tenir un servei**:

- Servidor «Tomcat» d'aplicacions, on mitjançant un script amb «bash» que s'encarrega de compilada l'API i copiar-la dins el contenidor `.war` al servei «Tomcat», i s'iniciara el container amb el Dockerfile.
- Base de dades ElasticSearch® distribuïda on disposarem d'un port obert, el qual estarà restringit sols per l'API o porta d'enllaç, mitjançant un «web service» (Apache R-proxy) que ens farà de servidor intermedi. Hi ha la informació per arrancar el container en README.md

Finalment, l'IDE o entorn integrat de desenvolupament per als repositoris de la porta de enllaç i Docker, serà *IntelliJ IDEA Community Edition* programari «open-source».

Repositori: https://github.com/acostasg/Docker

7. Desenvolupament

Durant la fase de desenvolupament i execució de les tasques (amb metodologia Kanban[1]), que s'han creat a partir del casos d'us, podem descriure i explicar el codi font de l'aplicació.

7.1 Descripció de l'aplicació

L'arquitectura de l'aplicació és una aplicació mòbil (**https://github.com/acostasg/POPmessage**), client-servidor, on disposem d'un programa java (Jersey & Jackson) que efectuara de servidor, és una API com a porta d'enllaç (**https://github.com/acostasg/API-Gateway-POPmessage**), que subministrara tant la informació necessària, així com el punts d'entrada per poder publicar aquests missatges, directori API-Gateway. La API utilitzara una serie de comandes (command pattern) de lectura/escriptura (**/API-Gateway/src/main/java/api/domain/command)**.

Les comandes utilitzaran la interfases del repositoris d'accés a dades «*repository pattern*» (**/API-Gateway/src/main/java/api/domain/infrastructure**).

En resum tenim el domini amb les entitats, commandes i interfícies en el package: **/API-Gateway/src/main/java/api/domain/**

Pel que fa les implementacions d'aquestes interfases les quals usa el domini de l'aplicació, arquitectura hexagonal, en el «package»: **/API-Gateway/src/main/java/api/infrastucture/**

Aquesta informació es persistir-ha en una base de dades documental (directori /**Docker/ElasticSearchCluster/elasticsearch**).

Són els repositoris les classes que «sabem» com usar *Elasticseach* i estan en el paquet **/API-Gateway/src/main/java/api/infrastucture/elasticSearch,** la resta del codi li és indiferent la base de dades utilitzada ja que utilitzen els contractes (interfícies) per comunicar-se amb els repositoris i aquest a la BBDD. (**https://github.com/acostasg/Docker**)

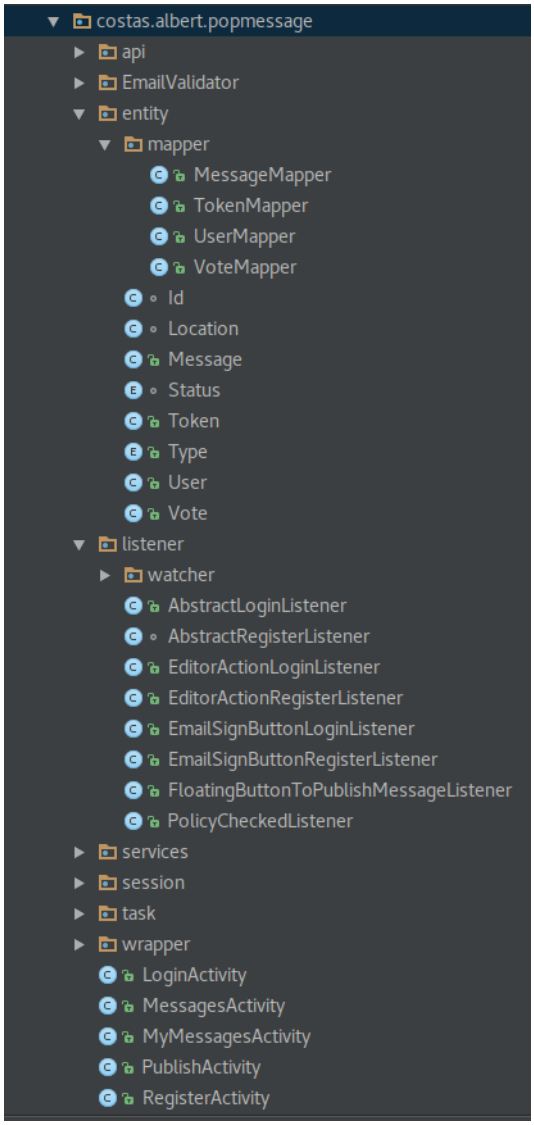
[1] **Metodologia Kanban:** Kanban és un mètode per gestionar el treball intel·lectual, amb èmfasi en el lliurament just a temps, mentre no se sobrecarreguin els membres de l'equip. En aquest enfocament, el procés, des de la definició d'una tasca fins al seu lliurament al client, es mostra perquè els participants ho vegin i els membres de l'equip prenguin el treball d'una cua.

[https://es.wikipedia.org/wiki/Kanban_(desarrollo)]

7.2 Arquitectura

7.2.1 Capa de presentació, repositori Android:

En el repositori d'Android,**https://github.com/acostasg/POPmessage**, hi ha el codi de l'aplicació Android, les classes on hi ha la lògica està implementades en el «*package*» **costas.albert.popmessage**:



costas.albert.popmessage.api: en aquest paquet hi ha tota les constants i encapsulament per la realització de peticions per l'api, podríem definir com les classes que «sabem» com comunicar-se amb la API.

costas.albert.popmessage.entity: son les presentacions de les entitats de domini, així com els «**mappers» o classes traductores**, que mitjançant un «decorador» amb les entitats, serialitzen i desserialitzen de format JSON a entitat i viceversa. En tot cas si canvies el format sols han de canviar les classes «*mappers*».

costas.albert.popmessage.listener: es la encapsulació de events o acciones que es produiran als components de les vistes com a botons, llistes, etc... per ser mes re-usables.

costas.albert.popmessage.services: utilitats de l'aplicació per homogeneïtzar i re-usar codi en tota l'aplicació: filtrar missatges, per agregar massatges als llistats, etc...

costas.albert.popmessage.session: és un paquet important, on és gestionen les dades d'usuari en la actual sessió: nom, «token» de sessió, etc...

costas.albert.popmessage.task: processos asíncrons, alguns relatius a les activitats o vistes del programari, que realitzen tasques com sol·licitar al «*package*» o «*classes*» de l'api missatges, registre d'usuari, etc... (sols tenen coneixement de entitats de domini, són les classes del package **costas.albert.popmessage.api** les que és comuniquen externament) un cop obtingut la resposta realitzen el canvis adequats a les vistes o informen als usuaris,.

costas.albert.popmessage.wrapper: Encapsulació de processos de transformació estàtics (realitzen accions sense guardar estats), com encriptació, ofuscació dels missatges, etc... sols esperen un paràmetres de entrada i donen un sortida a partir d'aquests, no fan absolutament res mes.

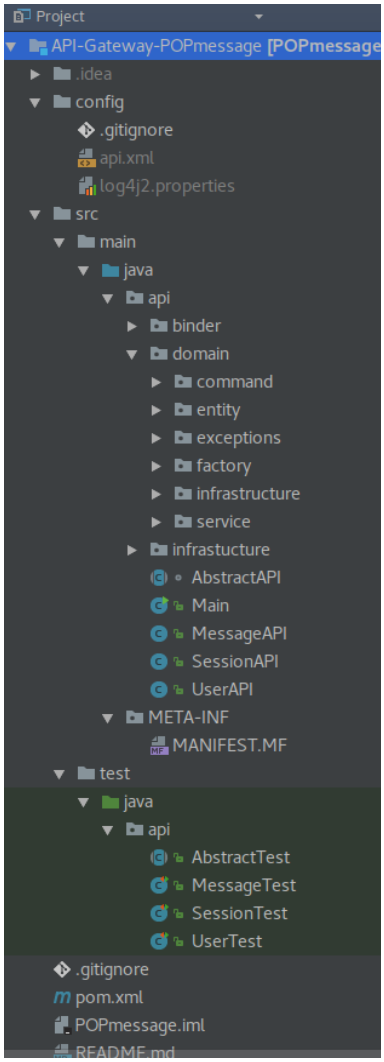
Activity són els «controladors» dels layouts o vistes, s'encarreguen de renderitzar els layouts que hi ha en el directori **/res** i els seus recursos, i delegar en els paquets anteriors diferents responsabilitats.

Pel que fa la resta es igual que qualsevol aplicació Android, amb el seu manifest, on hi ha el recursos que es demanaran a l'usuari del dispositiu (geolocalització), així com en la carpeta **/res** els fitxers amb extensió xml's de recursos: layouts, imatges vectorials, colors, etc...

7.2.2 Capa de control, repositori **API-Gateway**:

Servidor privat virtual: http://zonamessage.com:8080/application.wadl

https://github.com/acostasg/API-Gateway-POPmessage

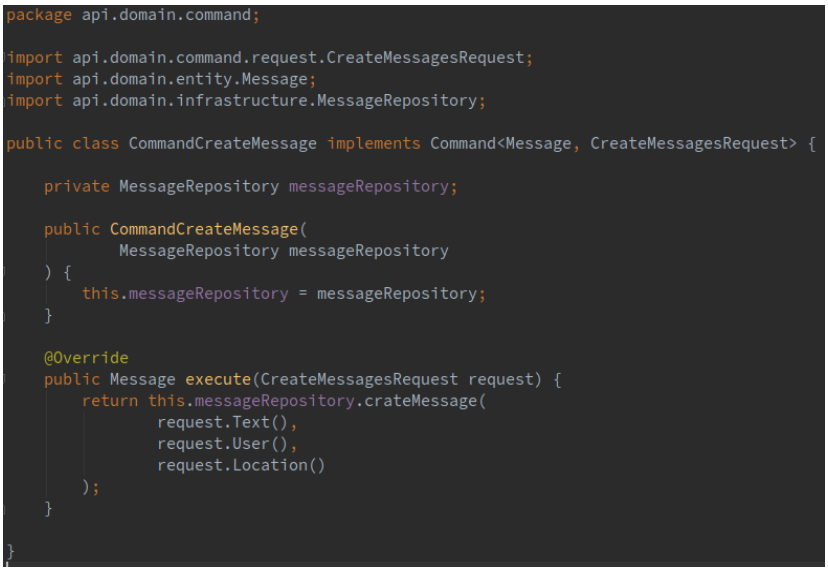


API de porta d'enllaç encarregada d'autenticació d'usuari i talla focs per accedir aquesta capa. Aquesta accedira a una serie de **comandes (command pattern)** per llegir/escriure mitjançat **interfases de repositoris (repository pattern)** que donaran accés a la **capa de persistència**, aquest fet en garanteix poder canviar de base de dades en un futur.

- **/config**
  - configuració del log4j i el fitxer amb format xml de configuració de dades de connexió, frases d'encriptació, keyApp, etc... aquests no han de ser públics i s'han de canviar en entorns productius.
- **/src/main/java/api/**
  - Es el paquet de la API, on hi ha el manifest que s'utilitza per crear el jar, així com classes amb «decorador» per al servidor Grizzly HTTP JAX-RS: MessageAPI, UserAPI i SessionAPI
  - Hi ha 2 capes, la capa de domini de aplicació **/src/main/java/api/domain** on està tota la

lògica i les entitats de domini que usen interfícies (contractes) per utilitzar l'infraestructura.

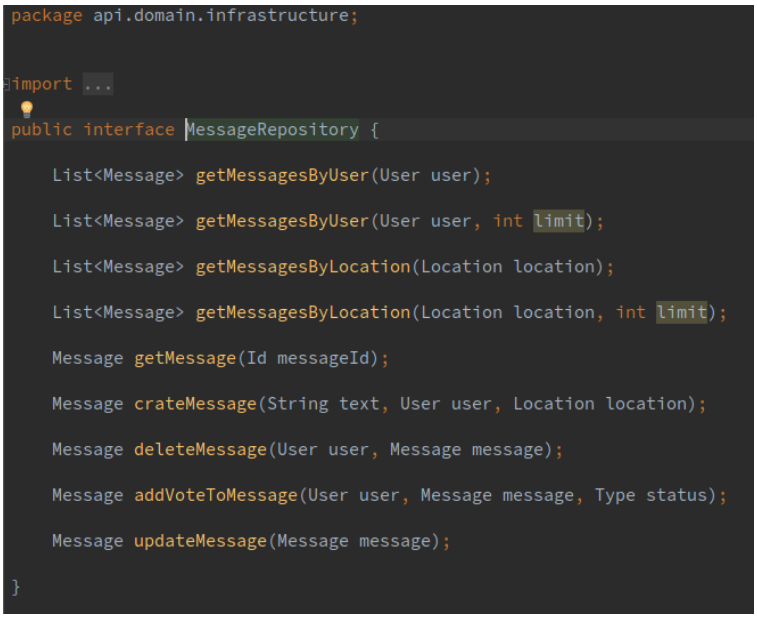
- **/src/main/java/api/domain/command,** on hi han les comandes que tenen la lògica de negoci, la entrada dades aquestes comandes des de la api es realitza amb objectes de petició modelitzats perquè estiguin validades les dades abans de la seva execució.



```
package api.domain.command;

import api.domain.command.request.CreateMessagesRequest;
import api.domain.entity.Message;
import api.domain.infrastructure.MessageRepository;

public class CommandCreateMessage implements Command<Message, CreateMessagesRequest> {

    private MessageRepository messageRepository;

    public CommandCreateMessage(
            MessageRepository messageRepository
    ) {
        this.messageRepository = messageRepository;
    }

    @Override
    public Message execute(CreateMessagesRequest request) {
        return this.messageRepository.crateMessage(
                request.Text(),
                request.User(),
                request.Location()
        );
    }

}
```

- **/src/main/java/api/domain/factory** factories (factory pattern) que tenen la responsabilitats de crear entitats de domini correctes amb les dades consistents.

- **/src/main/java/api/domain/infrastucture, interfícies** o **contractes** que ha de implementar l'infraestructura siguin quina sigui la base de dades, o tecnologia usada.



```
package api.domain.infrastructure;

import ...
public interface MessageRepository {

    List<Message> getMessagesByUser(User user);

    List<Message> getMessagesByUser(User user, int limit);

    List<Message> getMessagesByLocation(Location location);

    List<Message> getMessagesByLocation(Location location, int limit);

    Message getMessage(Id messageId);

    Message crateMessage(String text, User user, Location location);

    Message deleteMessage(User user, Message message);

    Message addVoteToMessage(User user, Message message, Type status);

    Message updateMessage(Message message);

}
```

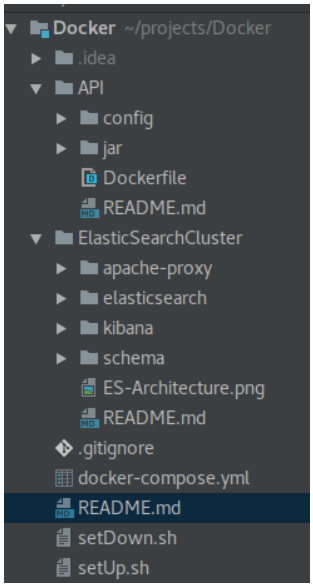
- **/src/main/java/api/domain/services** utilitats que es fan servir per tot el domini per homogeneïtzar i re-usar el codi: format de les dates, validadors, etc...

- **/src/main/java/api/infrastructure** on estan les implementacions respectives de les interfícies de domini per accedir a Elasticseach® o InMemory amb dades «falses» per als test.
  - **/src/main/java/api/infrastructure/elastisearch** son les classes que implementen les interfases de domini per comunicar aquest amb Elasticsearch.
  - **/src/main/java/api/infrastructure/InMemory** son les classes que implementen la interfícies de domini per proveir d'entitats de domini falses per al test
  - **/src/main/java/api/infrastructure/cache** son classes que implement interfícies de domini per tenir una petita memòria de 10 megabytes (amb ordre primer a entrar, primer a sortir) en memòria d'accés ràpid per evitar la consistència lleu i la repetició de processos repetitius.

- Destacar **/src/java/api/binder**, on es registren les interfícies o classes que s'utilitzen per inversió de control. El registre del servidor injecta les classes de infraestructura a partir de les interfícies, es podria canviar la implementació d'aquestes dinàmicament per qualsevol altra implementació de base de dades i la resta de codi no li afectaria.

7.2.3 Capa de persistència i entorn d'aprovisionament, repositori **Docker**:

https://github.com/acostasg/Docker



La base de dades reflecteix el model conceptual i operatiu on els hi ha usuaris, missatges i token de sessió, són els 3 indexs que s'han creat a *Elasticsearch*.

Podem veure en el directori **/Docker** com es realitza el aprovisionament de la base de dades en el fitxer de l'arrel **docker-compose.yml** es veu clarament com es crear el container de la API el qual es l'únic que te el port public 8080 d'accés, i la resta està en un clúster de *docker* amb xarxa privada que es comuniquen internament: API amb *Apache* (es podria aplicar un certificat per sols deixar passar peticions des

de API o de la futura aplicació de manteniment del administrador) i aquest amb el *ElastichSearch*.

En la següent imatge podem veure els contenidors en memòria:

```
→ Docker git:(master) docker ps
CONTAINER ID        IMAGE                                COMMAND                  CREATED             STATUS              PORTS                    NAMES
0b3dee57a3e5        popmessageapi                        "/usr/bin/java -ja..."   23 seconds ago      Up 22 seconds       0.0.0.0:8080->8080/tcp   popmessageapi
d4f744a51cc2        apache-proxy:1.0                     "apachectl -D FORE..."   24 seconds ago      Up 22 seconds       5000/tcp                 apache-proxy
15133bcc6fcd        popmessage-elasticsearch:2.4.1       "/usr/share/elasti..."   11 days ago         Up 22 seconds       9200-9300/tcp            es-node_3
b881d77ffb9d        popmessage-elasticsearch:2.4.1       "/usr/share/elasti..."   11 days ago         Up 22 seconds       9200-9300/tcp            es-node_1
091d3054a945        popmessage-elasticsearch:2.4.1       "/usr/share/elasti..."   11 days ago         Up 22 seconds       9200-9300/tcp            es-node_2
e456f4285df5        popmessage-elasticsearch:2.4.1       "/usr/share/elasti..."   11 days ago         Up 23 seconds       9200-9300/tcp            es-master
```

L'aprovisionament està documentat amb el **README.md** que hi ha en cada directori de *docker*, en tot cas podem veure que es realitzar als fitxers **Dockerfile**.

Es tan fàcil com utilitzar els script de bash creats per iniciar o parar el entorn:

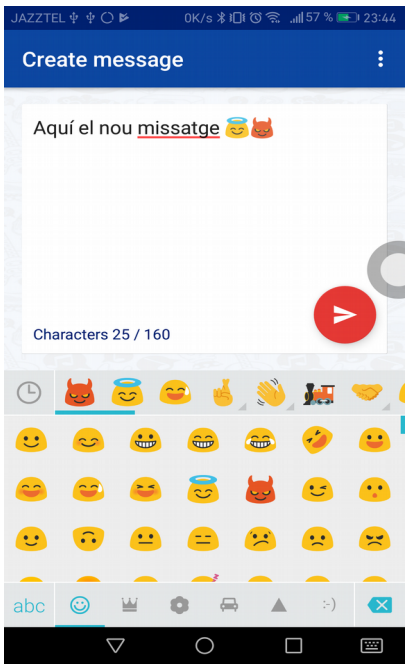
./setUp.sh
o
./setDown.sh

8. Funcionalitats

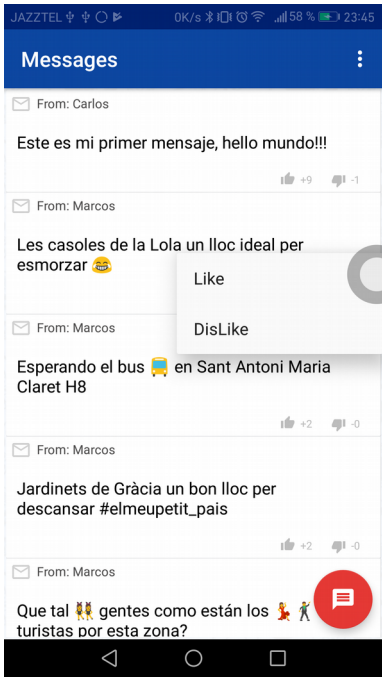
S'han implement les funcionalitats de les histories d'usuari i dels casos d'us.

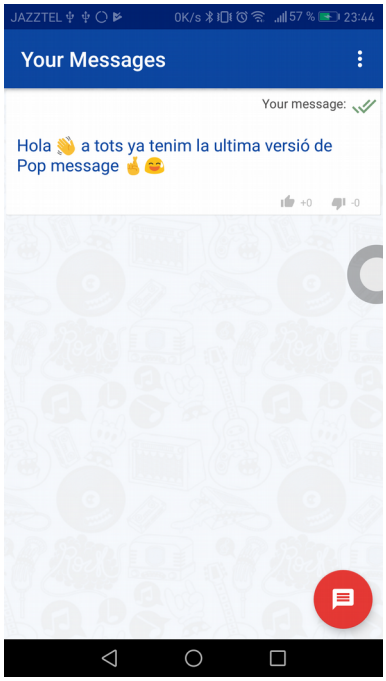
8.1.1 Usuari:

- Un usuari pot publicar un missatge en el punt geogràfic actual.
- Un usuari pot veure els 10 missatges de 5km a la rodona ordenats per vots positius, o ampliant el radi d'acció geogràfica fins veure 10 missatges.

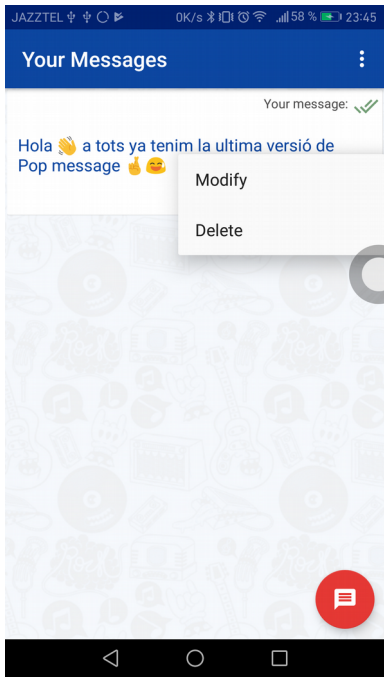


- Un usuari pot votar positivament o negativament un missatge.
- Un usuari pot veure els missatges publicat anteriorment.



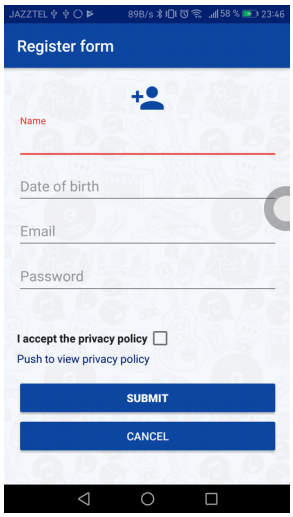


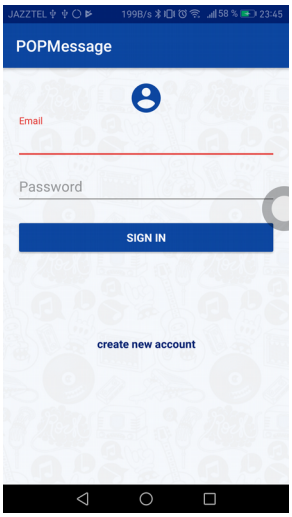
- Un usuari pot eliminar o modificar un missatge publicat anteriorment.



8.1.2 Administrador

- Tots els usuaris s'autentifiquen, el primer cop s'han de registrar:



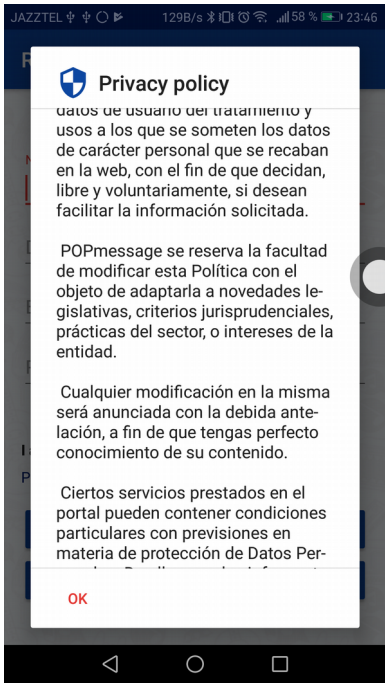


- El sistema es d'alta disponibilitat per a que l'usuari pugui utilitzar-lo en qualsevol moment.

  Per aquest motiu s'utilitza un clúster per garantir alta disponibilitat, /docker/ElasticsearchCluster

- ... bloquejar o desbloquejar missatges.
  - Aquests tenen un camp «*status*», si no es ACTIU no es mostra en l'aplicació.
- ... poder bloquejar o desbloquejar usuaris.
  - Aquests tenen un camp «*status*»s, si no es ACTIU no es mostra en l'aplicació.
- ... eliminar missatges.
  - Aquests tenen un camp «*status*», si no es ACTIU no es mostra en l'aplicació, es pot realitzar pel mateix usuari que a realitzat el missatge de la APP o administrador del sistema.
- ... eliminar usuaris.
  - No s'ha implementat que els mateixos usuari es pugin esborrar, s'informa a la política de privacitat d'un correu electrònic per poder donar-se de baixa, tot i que si no tenen el camp «*status*» ACTIU no es mostren els seus missatges.
- La base de dades informa de registres amb la seva geolocalització.
  - És gestiona des de la base de dades, en aquest cas ElasticSearch® amb els repositoris corresponents.

8.1.3 Administració pública



- S'informe als usuaris del seus drets durant el registre de les seves dades.
- S'indiqui l'acotació i el propòsit de l'aplicació i l'ús de les dades privades del usuari.
- L'usuari pot en tot moment sol·licitar la eliminació de les seves dades
    - En el moment de registre es poden veure les polítiques de privacitat i és descriu com l'usuari es pot donar de baixa.

8.1.4 Desenvolupador

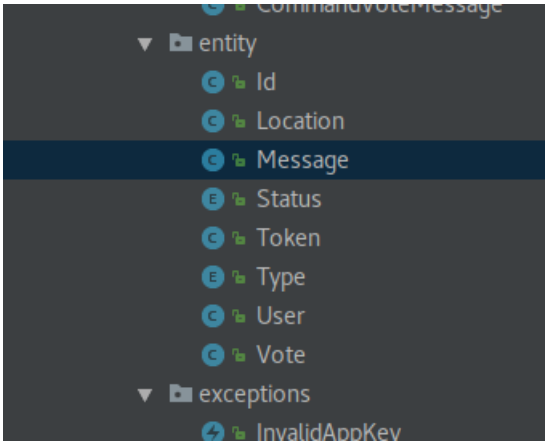
- S'ha aprovisionat l'entorn d'una forma automatitzada.

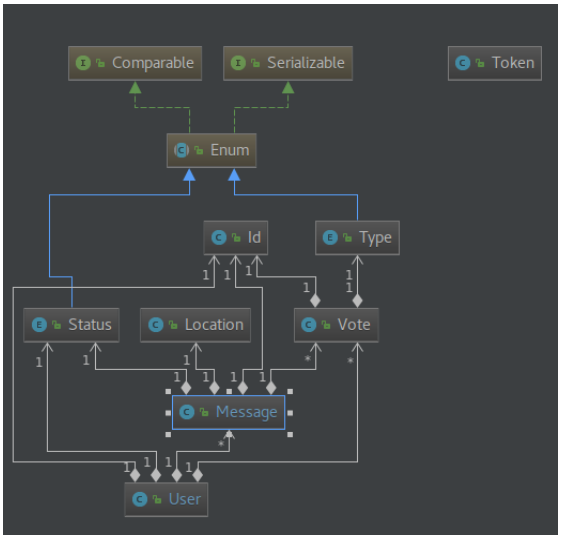
S'ha realitza't l'automatització de l'aprovisionament amb *docker-compose*, és pot veure en el /Docker/docker-compose.yml. Veure el punt 7.2.3.

Repositori: https://github.com/acostasg/Docker

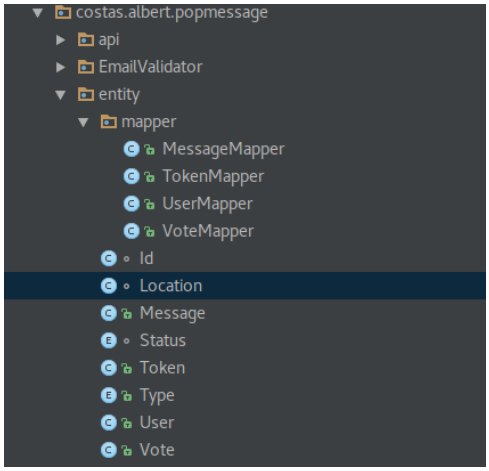
9. Implementació entitats

Pel que fa les entitats de domini, o la seva representació, en la **API** estan en «*package*» **api.domain.entity:**





També hi ha la seva representació en la aplicació **Android®** , que mitjançat «*mappers*» o *classes «traductores»* per la de-serialització (s'utilitza un decorador @JacksonAnnotation) en el transport amb format JSON, son iguals i estan a «*package*» **costas.albert.popmessage.entity**:



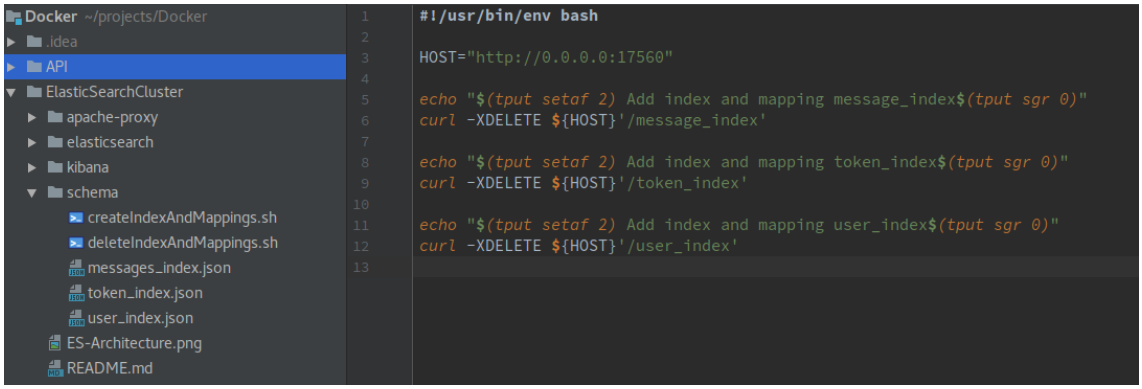
9.1 ElasticSearch

S'ha seleccionat *ElasticSearch®* , una base de dades documental, distribuïda, escalable i d'alta disponibilitat. S'utilitza un clúster de nodes per tenir alta disponibilitat, per contra tenim un consistència lleu. En el punt 7.2.3 podem veure els contenidors de *Docker* on està cada node.

- Es pot veure la estructura de dades en el directori **/Docker/ElasticSearchCluster/Schema** on estan la definició dels 3 index **message, user i token.**

- Així com els scripts en *bash* per crear aquesta estructura en *Elasticsearch®* , un cop aquesta està disponible per primer cop:

    - **./createIndexAndMappings.sh**



Script amb bash per la creació del index i mappers de les entitats a la base de dades documental ElasticSearch.

10. Funcionalitats per a properes versions

Pel que fa les funcionalitats que no es realitzaran en aquest projecte, tot i que, s'espera en properes versions:

- La interfície de l'usuari multi-idioma, on l'usuari pugui disposar de l'aplicació en l'idioma escollit.
- Que el missatges tinguin etiquetes (tags) per classificar els missatges i poder filtrar.
- Possibilitat de recordar la contrasenya a l'usuari.
- Veure una simulació geogràfica (Google maps©) de la disposició/distribució del missatges que envolten la posició actual de l'usuari.
- Integració de publicitat intel·ligent mitjançant informació de missatges (Google adWords) amb la geoposició de l'usuari (es podria fer una integració amb la API de Google Maps) i aplicació de *Deep learning or Machine learning* quan hi hagi suficient informació.
- Panel d'administració perquè els rols d'administrador puguin bloquejar tant missatges com usuaris, aquesta gestió sols es podria accedir via VPN.
- Utilitzar SSL entre API i APP amb un certificat auto firmat o d'una CA [https://developer.android.com/training/articles/security-ssl.html#HttpsExample]
- Enviar una notificació als usuari quan es vota positivament el seu missatge o negativament [https://www.adictosaltrabajo.com/tutoriales/configurando-notificaciones-push-android/#03]

11. Conclusió

En primer lloc, el plantejament inicial del project l'he afrontat com un repte tecnològic i la oportunitat d'aprendre en l'àmbit de les noves tecnologies en la àrea Java EE. Cal destacat del fet anterior, que no havia realitzar cap projecte enfocat específicament per a dispositius mòbils, el qual significava un risc però la motivació per l'aprenentatge.

Per tant, en els primers compassos del treball era molt important la planificació, la investigació i l'aprenentatge del SDK de Android i les seves bones practiques. Sense deixar de banda tota la gestió de l'API amb JAVA i les necessitat d'una aplicació d'alt rendiment, com potser una aplicació de missatgeria.

Per aquest fet he de destacar algunes assignatures que ha sigut clau per poder afrontar aquest projecte:

- Sistemes distribuïts i disseny de base de dades com a part fonamental d'un sistema d'altes prestacions
- Disseny i programació, anàlisis i disseny patrons, i interacció persona ordinador entre altres pel que fa el disseny, arquitectura i implementació de l'aplicació.
- Enginyeria de requisits i competències comunicatives pel que fa d'utilització, adequació i correcció dels textos realitzats.

Pel que fa els objectius inicials plantejats en aquest document, a més del factor de aprofundir amb el coneixement adquirits durant el grau, estaven definit i limitats dins del context del treball final de grau, és a dir, s'ha centrar el focus en un desenvolupament realista amb el temps i recursos. D'altra banda, s'ha deixar la porta oberta a futures millores.

Fet el qual, sumat amb la feina i la metodologia utilitzada considero que s'han complet els objectius favorablement, tot i que, durant l'execució del projecte han sortit possibles millores i noves funcionalitats interessants per una futura ampliació, per realització personal.

El plantejament de la feina durant l'execució de l'aplicació ha sigut un model seqüencial en cascada, degut al únic recurs disponible. Tot i això, crec que la realització dels casos d'ús i la creació de les histories ha sigut un enfocament BDD (*Behavior-driven development*), i la seva execució ha pogut ser realitzada en metodologia àgil Kanban, tot i que no d'una forma estrica.

Com a claus d'èxit podem veure, primer, amb la finalització dels casos d'ús, d'una forma estable i amb test en l'API. I segon, en l'aplicació dels patrons i arquitectura que s'havia especifica en el disseny, fet que es demostrar amb una arquitectura hexagonal

on la lògica de domini està separa de la infraestructura. A més, ens permetria tenir més d'una base de dades o canviar aquesta sense impacte en el nostre domini.

La utilització d'una API ens permetria tenir no sols una aplicació d'Android, sinó per exemple un altre programari amb iOS o una pagina Web consumin la nostra API sense que aquesta tingues cap tipus de modificació.

En definitiva, el treball final de grau ha sigut un repte amb el seu component de risc i de realització personal. La planificació, l'aprenentatge i el coneixement adquirit durant el grau han aconseguit la realització d'una aplicació d'alt rendiment i la consumació d'un projecte. Amb la satisfacció d'haver usat una arquitectura que permet la millora i el manteniment del programari durant el seu cicle de vida.

12. Glossari

Interconnexió	Acció d'interconnectar, en el actual context connexió entre dos o més persones mitjançat les xarxes informàtiques.
Xarxes socials	Una xarxa social és una estructura social composta per individus (o organitzacions) anomenats "nodes" que estan lligats (connectats) per un o més tipus d'interdependència com ara amistat, parentesc, interessos comuns, ... [https://ca.wikipedia.org/wiki/Xarxa_social]
Arquitectura de l'aplicació	Referent a un programari, és el disseny conceptual i l'estructura operacional fonamental d'un sistema. És a dir, és un model i una descripció funcional dels requeriments i les implementacions de disseny per diverses parts del programari.
API	Una interfície de programació d'aplicacions (en anglès Application Programming Interface, API) és una interfície que especifica com diferents components de programes informàtics haurien d'interaccionar. [https://ca.wikipedia.org/wiki/Interf%C3%ADcie_de_programaci%C3%B3_d'aplicacions]
Grafs	En teoria de grafs, un graf és una representació abstracta d'un conjunt d'objectes on alguns parells dels objectes estan connectats per enllaços. Els objectes interconnectats són representats per abstraccions matemàtiques anomenades vèrtexs, i els enllaços que connecten alguns parells de vèrtexs s'anomenen arestes. [https://ca.wikipedia.org/wiki/Graf_(matem%C3%A0tiques)]
BDD	És un procés de desenvolupament programari, si bé es relaciona principalment amb el Testing, que és d'on sorgeix. BDD busca un llenguatge comú per unir la part tècnica i la de negoci, i que sigui des d'aquest llenguatge comú des d'on arrenqui el Testing. [http://www.javiergarzas.com/2014/12/bdd-behavior-driven-development-1.html]
Domini d'aplicació	En l'actual context, es referix al àrea temàtica o camp que l'usuari aplicara a l'aplicació, és a dir, representa la terminologia i el conceptes clau que definixen l'aplicació.
Talla focs	Un tallafoc és un element de maquinari o programari utilitzat en una xarxa d'equips

	informàtics per controlar les comunicacions, permetent-les o prohibint-les segons les polítiques de xarxa que hagi definit l'organització responsable de la xarxa. [https://ca.wikipedia.org/wiki/Tallafoc_(inform%C3%A0tica)]
Aplicació escalabre	És la propietat desitjable d'un sistema, una xarxa informàtica o un procés, que indica la seva habilitat per a estendre el marge d'operacions sense perdre qualitat. [https://ca.wikipedia.org/wiki/Escalabilitat]
Teorema CAP	En informàtica teòrica, el teorema CAP, també conegut com a teorema de Brewer, formula que és impossible garantir simultàniament les tres característiques següents en una aplicació distribuïda: Consistència , Disponibilitat i Tolerància a la partició. [https://ca.wikipedia.org/wiki/Teorema_CAP]
Autenticació	És l'acte d'establiment o confirmació d'alguna cosa (o algú) com a autèntic.
Geolocalització	És la capacitat per obtenir la ubicació geogràfica real d'un objecte, com un radar, un telèfon mòbil o un ordinador connectat a Internet. [https://ca.wikipedia.org/wiki/Geolocalitzaci%C3%B3]
Metàfora	Una metàfora d'interfície consisteix a emprar una semblança o analogia per fer la interacció més simple i intuïtiva. [http://www.uxlumen.com/pestanas-metaforas-y-diseno-plano]
Patrons	És una solució general a un problema comú i recurrent en el disseny de programari. [https://ca.wikipedia.org/wiki/Patr%C3%B3_de_disseny]
Arquitectura hexagonal	L'Arquitectura Hexagonal promou la separació de preocupacions a través de capes de responsabilitat. Cada capa de l'aplicació té un estricte conjunt de responsabilitats i preocupacions. [http://codely.tv/screencasts/arquitectura-hexagonal-ddd/]
API *stateless*	Aplicacions que no mantenen un estat. [https://carlosazaustre.es/blog/que-es-la-autenticacion-con-token]

13. Bibliografia i Enllaços de interès

[1]. Command Pattern: https://en.wikipedia.org/wiki/Command_pattern

[2]. BDD – Guida Agilea Alliance: http://guide.agilealliance.org/guide/bdd.html

[3]. IntelliJ IDEA : https://www.jetbrains.com/idea/

[4]. Android® : https://developer.android.com/studio/index.html

[5]. Interfaz de usuario: https://developer.android.com/guide/topics/ui/index.html

[6]. Patrones i interración moviles: http://appdesignbook.com/es/contenidos/patrones-interaccion-moviles/

[7]. Crea tu primera app: https://developer.android.com/training/basics/firstapp/index.html

[8]. Docker: https://www.docker.com/

[9]. Github: https://github.com/

[10]. Elasticsearch: https://www.elastic.co/

[11]. Api gateway: https://networknt.github.io/light-java/architecture/gateway/

[12]. Diccionari.cat: http://www.diccionari.cat/

[13]. Softcatalà sinònims: https://www.softcatala.org/diccionari-de-sinonims/

[14]. Wireframe: https://wireframe.cc/

[15]. Viquipèdia: https://ca.wikipedia.org/wiki/Portada

[16]. Iconic Font is a free set of icons in one font for your next project: http://iconmonstr.com/iconicfont/

[17]. Crear una campaña de AdWords para tu app: https://support.google.com/googleplay/android-developer/answer/6262700?hl=es-419

[18]. Arquitectura Android - Google : https://developer.android.com/guide/platform/index.html?hl=es